POLITECHNIKA KRAKOWSKA
IM. TADEUSZA KOŚCIUSZKI

Systemy Baz Danych

Projekt: Aplikacja wspomagająca pracę gabinetu kosmetycznego

Prowadzący - prof. Dr hab. Inż. Sergii Telenyk

Wydział	WIEiK	
Kierunek	Informatyka	
Zespół	Wąsik Hubert	33i
	Pawlikowska Anna	32i
	Piskorz Paweł	33i

Spis treści

1 Cel i zakres projektu

Celem projektu jest utworzenie aplikacji, której odbiorcą jest zarówno właściciel gabinetu kosmetycznego, jak i pracownik oraz klient. Aplikacja służy do zarządzania gabinetem kosmetycznym oraz umożliwia potencjalnym klientom w łatwy sposób przeglądać ofertę gabinetu, zapisać się na konkretny zabieg bądź zgłosić reklamację dotyczącą wykonanej usługi.

Aplikacja musi spełniać wypisane niżej funkcjonalności niezbędne do działania gabinetu kosmetycznego, a których zakres skonsultowany został z właścicielem gabinetu.

2 Charakterystyka użytkowników

Administrator (Właściciel) – odpowiedzialny jest za zarządzanie istniejącymi kontami, usługami udostępnianymi przez gabinet oraz zasobami gabinetu. Posiada dostęp do danych klientów i pracowników, gdzie ma możliwość ich edytowania.

Użytkownik (Pracownik, Klient) – konto posiadające dwa typy – pracownik bądź klient. W zależności od typu użytkownika posiada dostęp do określonych funkcji systemu, takich jak zapisywanie się na zabieg czy zgłaszanie reklamacji.

3 Główne funkcje produktu

1) Logowanie do systemu.
2) Przedstawianie zakresu usług gabinetu.
3) Wspomaganie zarządzania zasobami ludzkimi.
4) Wspomaganie zarządzania zasobami gabinetu typu produkty/urządzenia.
5) Umożliwienie komunikacji pomiędzy klientem a pracownikiem.
6) Wspomaganie kierowania gabinetem.

4 Wykorzystywane technologie

1) Java oraz JavaServer
2) MySQL
3) JavaScript
4) HTML oraz CSS
5) Bootstrap Framework

5 Wymagania funkcjonalne i niefunkcjonalne

5.1 Wymagania funkcjonalne

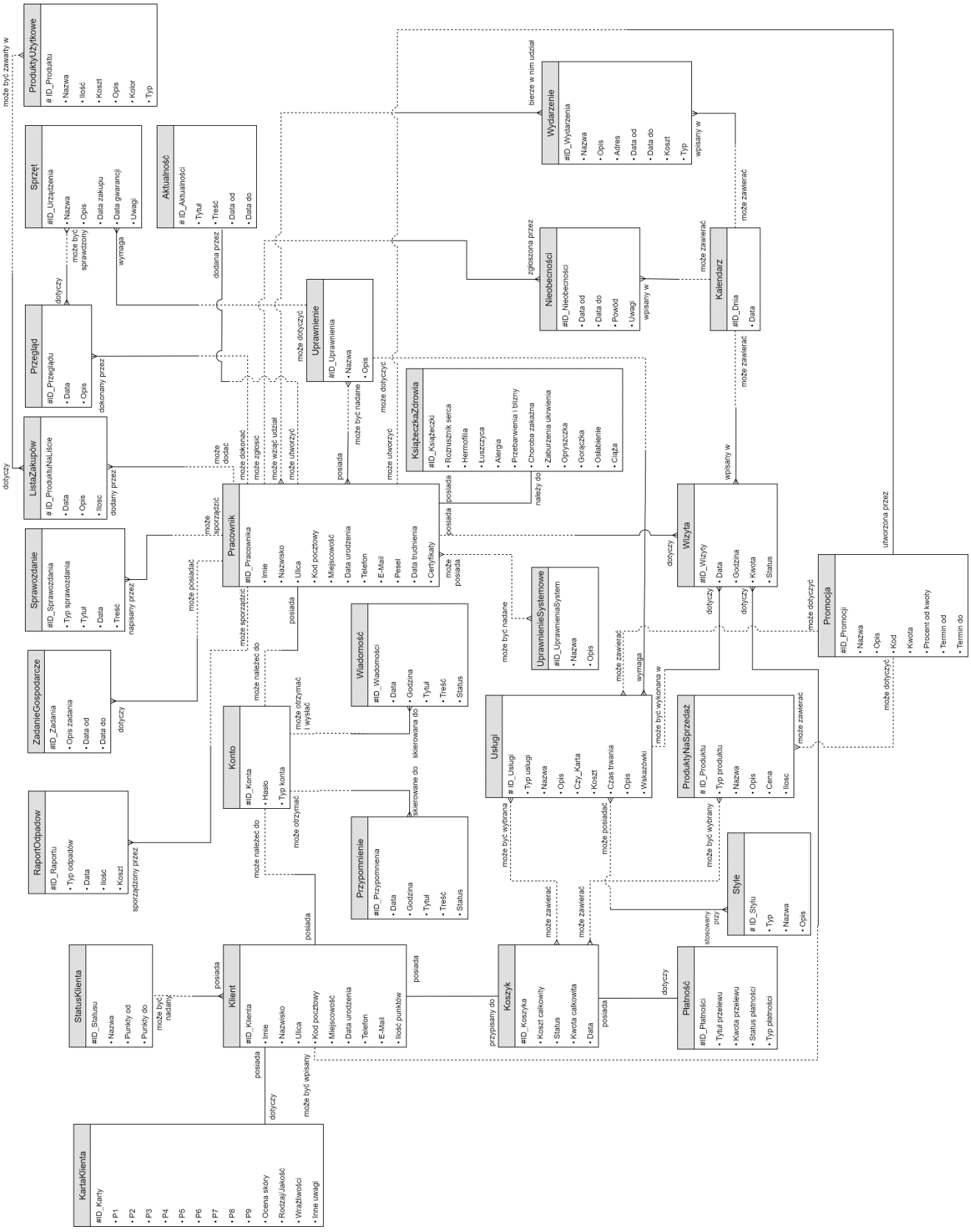
1) Logowanie i wylogowanie użytkowników – klient / administrator / pracownik.
2) Tworzenie kont pracowników, ich edycja oraz zarządzanie uprawnieniami (systemowymi jak i usługowymi) przez administratora.
3) Dodawanie przez administratora terminu szkoleń i kongresów kosmetycznych, informacji o ich zakresie, wyświetlanie ich w systemie oraz możliwość zapisu przez pracownika.
4) Dodawanie przez administratora bądź zalogowanego w gabinecie pracownika klienta wraz z zamówioną usługą do grafiku (kalendarza), możliwość jego edycji i podglądu.
5) Tworzenie „Kart Informacyjnych" zawierające wskazania i przeciwskazania do zabiegów oraz alergie i nietolerancje organizmu klienta możliwych do edytowania przez klienta oraz „Kart Zabiegowych" informujące o wykonywanych zabiegach u danego klienta.
6) Możliwość prowadzenia książeczki zdrowia pracowników.
7) Dodawanie zatwierdzonych osiągnięć – ukończone szkolenia oraz otrzymane certyfikaty – przez administratora do profilów pracowników oraz ich prezentacja.
8) Dodawanie sprawozdania zawierającego przebieg ostatnio wykonanego przeglądu technicznego (wewnętrzny), serwisowego (zewnętrzny) oraz sporale kontrolne (medyczne), ich daty przeprowadzenia oraz możliwość ustawienia przypomnienia.
9) Możliwość dodawania wybranych sprzętów użytkowych, informacji o nich oraz ich wyświetlanie.
10) Zarządzanie odpadami – utylizacja – generowanie sprawozdania odbioru zarejestrowanych odpadów (data oraz ilość), ustawienie okresowego przypomnienia o zbliżającym się terminie odbioru odpadów oraz generowanie rocznego bilansu utylizacji.
11) Zarządzanie stanem magazynowym materiałów kosmetycznych - możliwość zgłoszenia braków i propozycji zwiększające ofertę o nowości gabinetu przez pracownika oraz stworzenie listy zakupowej przez administratora na ich podstawie.
12) Rejestrowanie wizyt kontrolnych, takich jak sanepid, ZUS czy państwowa inspekcja pracy.
13) Możliwość generowania wykresu zawierającego ilość wykonanych poszczególnych usług w danym miesiącu.
14) Promowanie social-mediów gabinetu.
15) Dodawanie przez administratora bądź wyznaczonego pracownika informacji o aktualnych ofertach, pakietach sezonowych, okolicznościowych, świątecznych i konkursach oraz ich udostępnienie klientowi.
16) Prowadzenie programu motywacyjnego dla pracownika – nadawanie bonusów okolicznościowych i premii.
17) Możliwość prowadzenia spisu wykonanych usług, kto ich wykonał oraz kwota otrzymanej zapłaty.
18) Możliwość podglądu przez pracownika w swoim profilu informacji o wysokości nadchodzącej wypłaty.
19) Ustalanie z miesięcznym wyprzedzeniem grafiku oraz możliwość zgłaszania prośby o jego edycję przez pracownika.
20) Zgłaszanie prośby o przyznanie urlopu bądź poinformowanie o otrzymaniu zwolnienia lekarskiego przez pracownika administratorowi.
21) Możliwość wprowadzenia zakresu zadań gospodarczych poszczególnym pracownikom.
22) Dodawanie w systemie CV potencjalnego pracownika (w przypadku gdy poszukiwany jest pracownik) oraz przesłania go na mail administratora.
23) Możliwość konsultacji online poprzez chat online pomiędzy klientem a pracownikiem.
24) Możliwość przeglądania oferty usług udostępnianych przez gabinet oraz wybór wzorów/kolorów/typu makijażu przez klienta.

25) Możliwość wstępnej rezerwacji terminu wykonania wybranej usługi u wybranego pracownika.
26) Możliwość składania formularza reklamacyjnego.
27) Złożenie zamówienia przez klienta dotyczącego kupna produktu bądź usługi znajdującego się w ofercie gabinetu oraz opłacenia go przelewem bankowym.
28) Usługa lojalnościowa – klient otrzymuje status stałego klienta po skorzystaniu z określonej ilości usług zatwierdzonych przez pracownika. Takiemu klientowi przysługują bonusy określane przez właściciela gabinetu.
29) Automatyczne wysyłanie przypomnienia klientowi o wizycie (mail).
30) Informowanie klienta o procedurze przygotowania się do zabiegu.
31) Informowanie klienta o celu przetwarzania ich danych osobowych (RODO), przez kogo mogą być wyświetlane i kto nimi zarządza.

5.2 Wymagania niefunkcjonalne

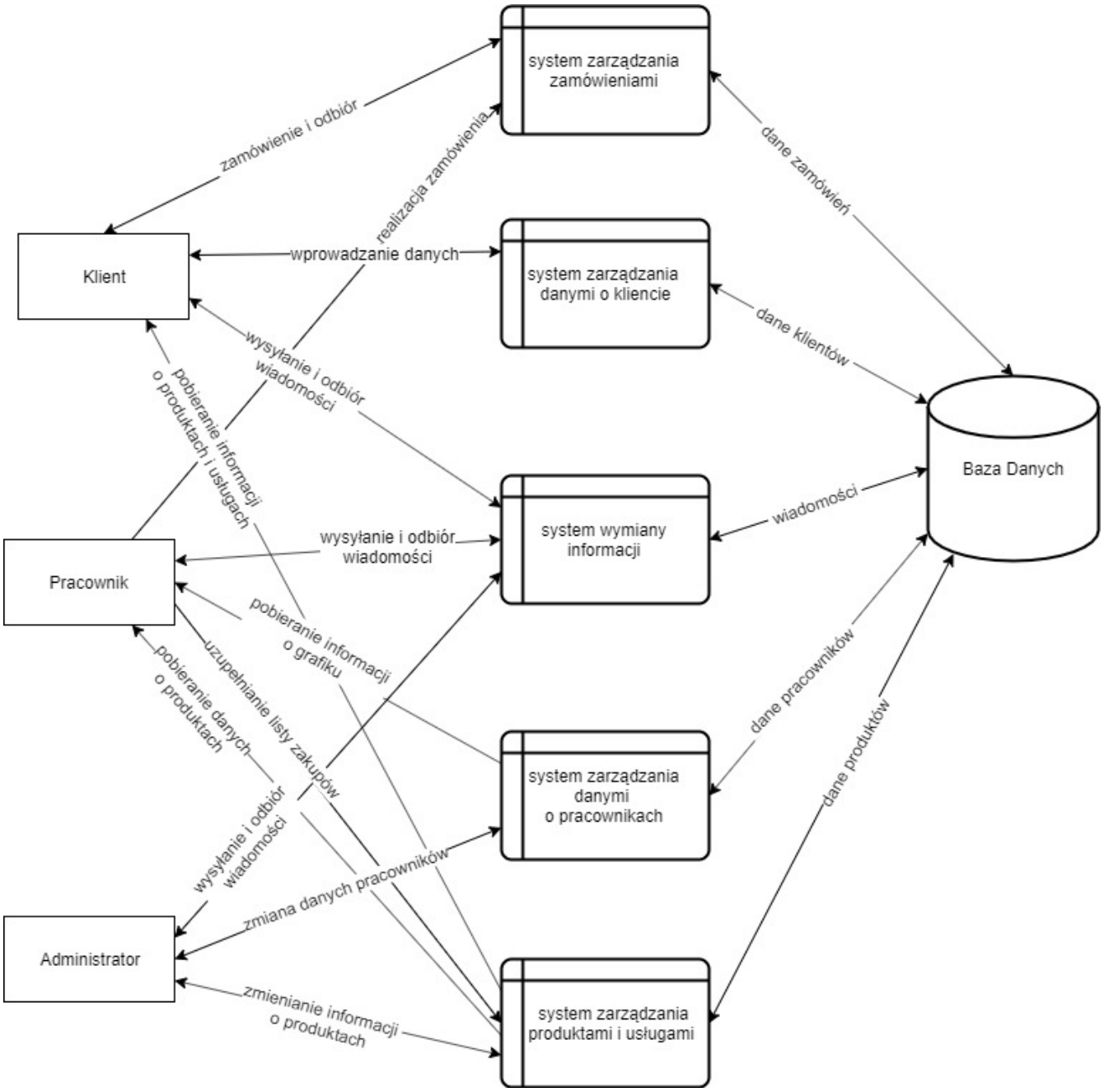
1) Aplikacja ma być niezawodna i bezbłędna, tzn. realizować swoje zadania w ściśle określonym przedziale czasu i robić to poprawnie.
2) Aplikacja ma być bezpieczna - rozumiemy przez to zabezpieczenia przed niepowołanym dostępem do aplikacji.
3) Aplikacja ma być niezależna od platformy - posiadać możliwość działania na każdym urządzeniu bez względu na to, na jakim systemie operacyjnym zostaje uruchomiona jak i posiadanej mocy obliczeniowej urządzenia.
4) Aplikacja ma być prosta w obsłudze - interfejs będzie intuicyjny, przejrzysty i czytelny.
5) Aplikacja ma stosować hierarchię dostępu - wynika to z podziału funkcjonalności: użytkownik zarejestrowany jako klient nie może mieć dostępu do funkcji administratora.
6) Aplikacja nie powinna udostępniać żadnych danych osobowych użytkowników osobom do tego nieuprawnionym.
7) Aplikacja nie może pozwalać na wprowadzanie istotnych zmian pracownikowi w systemie jeśli nie jest zalogowany za pośrednictwem sieci znajdującej się w gabinecie.

6 Diagram ERD przed normalizacją

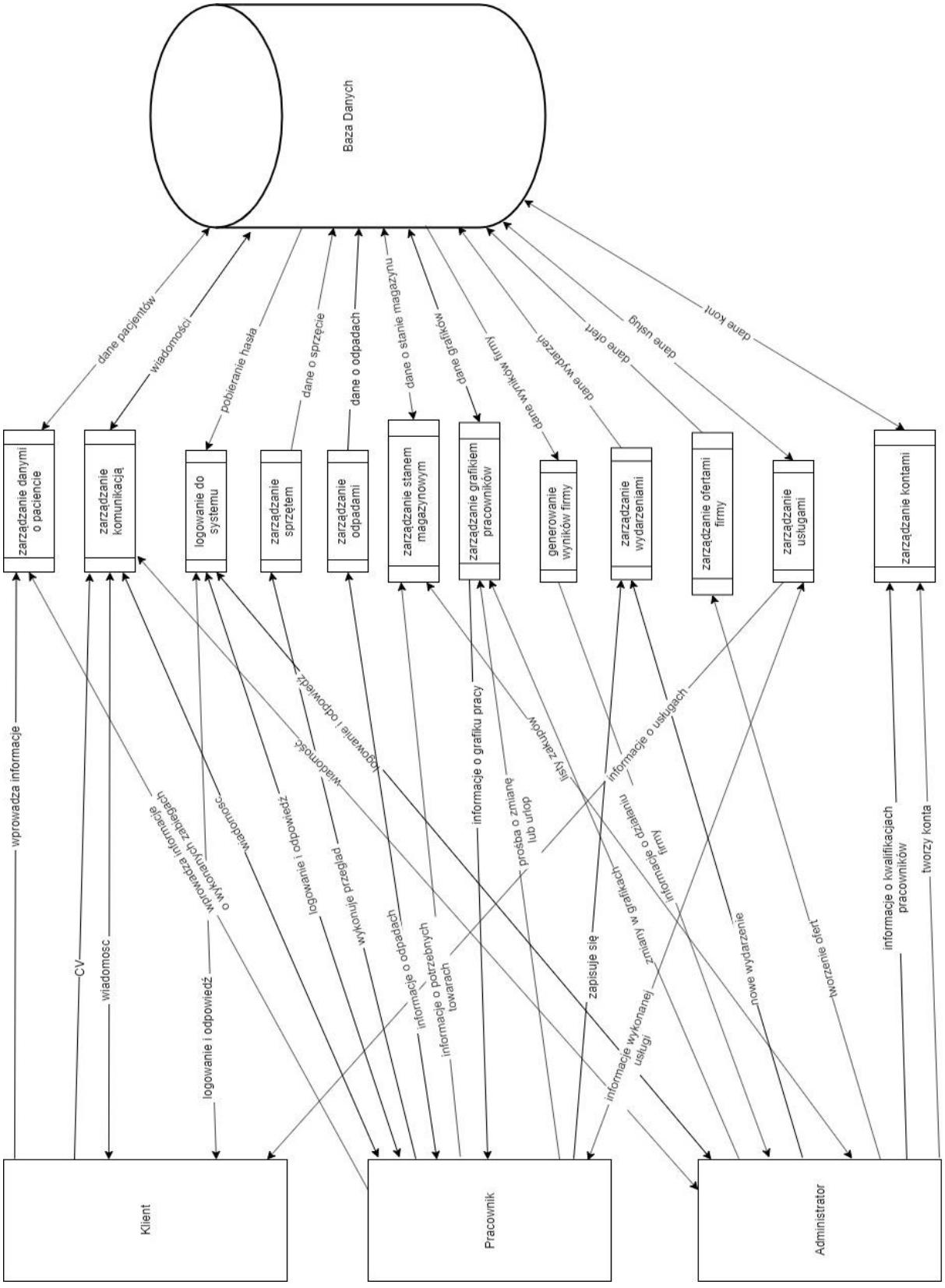


7 Diagramy DFD

7.1 Diagram kontekstowy



7.2 Diagram systemowy

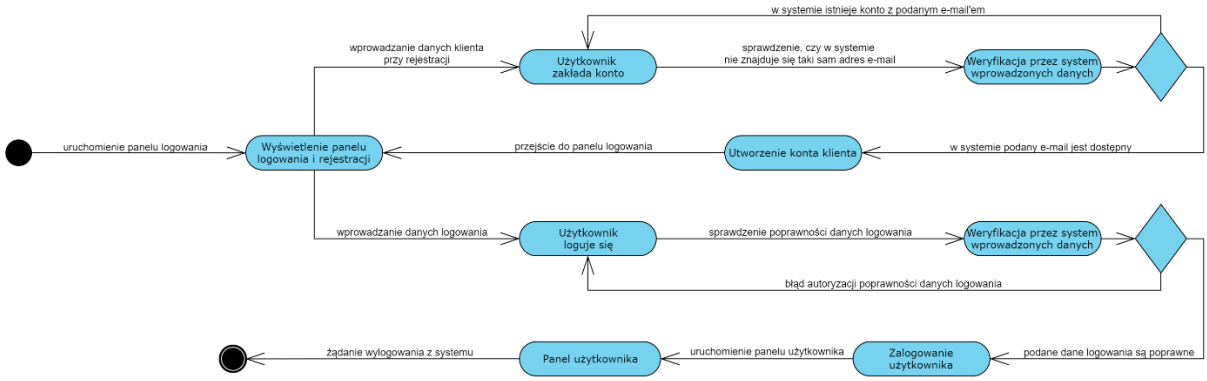


8 Diagramy STD

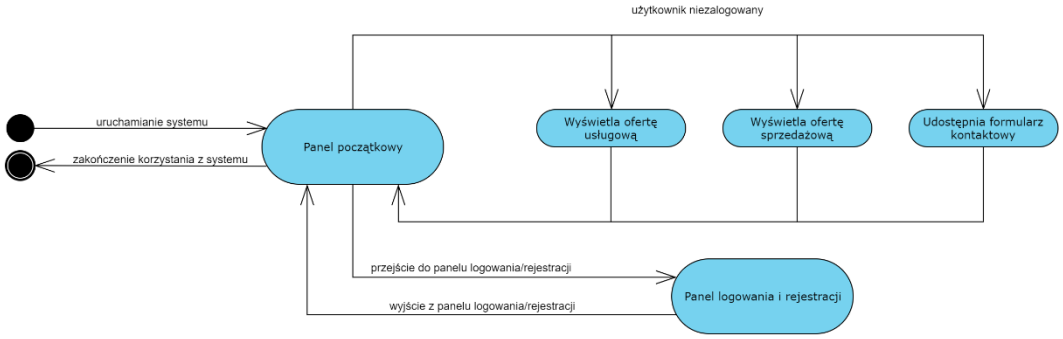
Poniżej zamieszczone zostały diagramy maszyn stanów dla poszczególnych funkcjonalności aplikacji:

- proces logowania użytkownika i rejestracji klienta
- proces dla osoby niezalogowanej w systemie
- proces dla zalogowanego w systemie klienta
- proces dla zalogowanego w systemie pracownika
- proces dla zalogowanego w systemie administratora

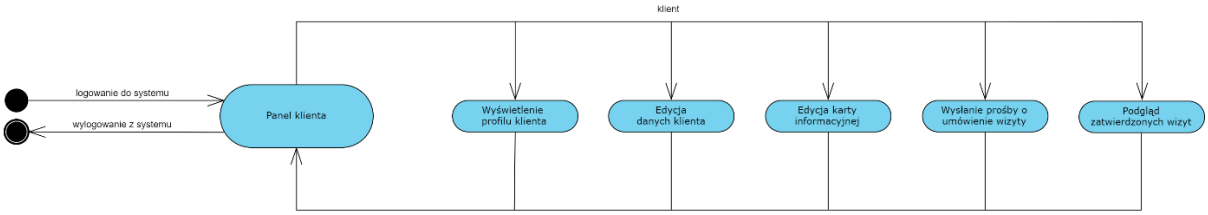
8.1 Proces logowania użytkownika i rejestracji klienta



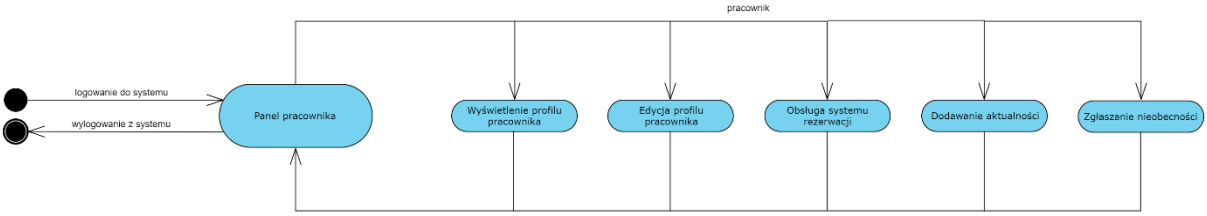
8.2 Proces dla osoby niezalogowanej



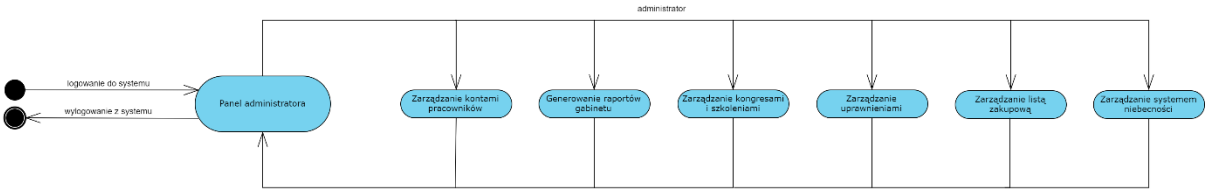
8.3 Proces zalogowanego klienta



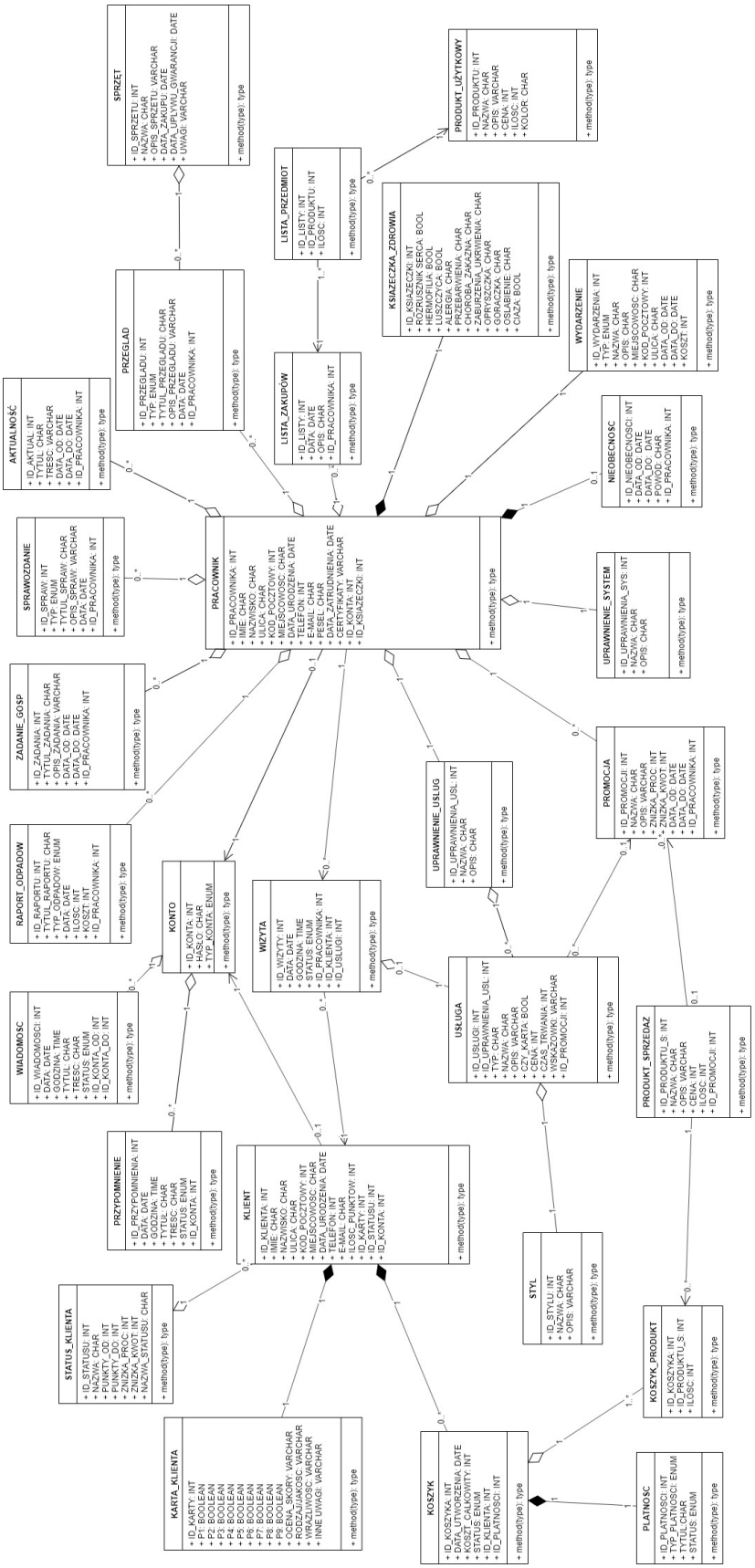
8.4 Proces zalogowanego pracownika



8.5 Proces zalogowanego administratora



9 Diagram klas



10 Baza danych

10.1 Proces normalizacji schematu bazy danych

Przedstawiony na pierwszym schemacie model bazy danych przekształciliśmy do 1NF, gdzie należało uwzględnić to, by każde pole przechowywało jedną informację oraz to, by każda encja mogła być identyfikowana za pośrednictwem własnego, oryginalnego i niepowtarzalnego klucza głównego.

W procesie normalizacji zauważyliśmy, iż:

- encja „kalendarz" jest encją zbędną i niepotrzebną, by spełnić jedno z założeń projektu jakim jest obsługa rezerwacji wizyt online. Zakładając, że w gabinecie każdy z pracowników powinien być obecny od godziny 10:00 do 18:00 (gabinet, dla którego tworzona jest aplikacja nie posiada zmian porannych/wieczorowych pracy pracowników) można w łatwy sposób stwierdzić, czy pracownik jest do dyspozycji w wybrany dzień o wybranej godzinie. Wystarczy jedynie sprawdzić, czy nie posiada wpisanej „nieobecności" (zatwierdzonego przez administratora systemu zwolnienia) oraz czy nie posiada o wybranej godzinie wizyty.

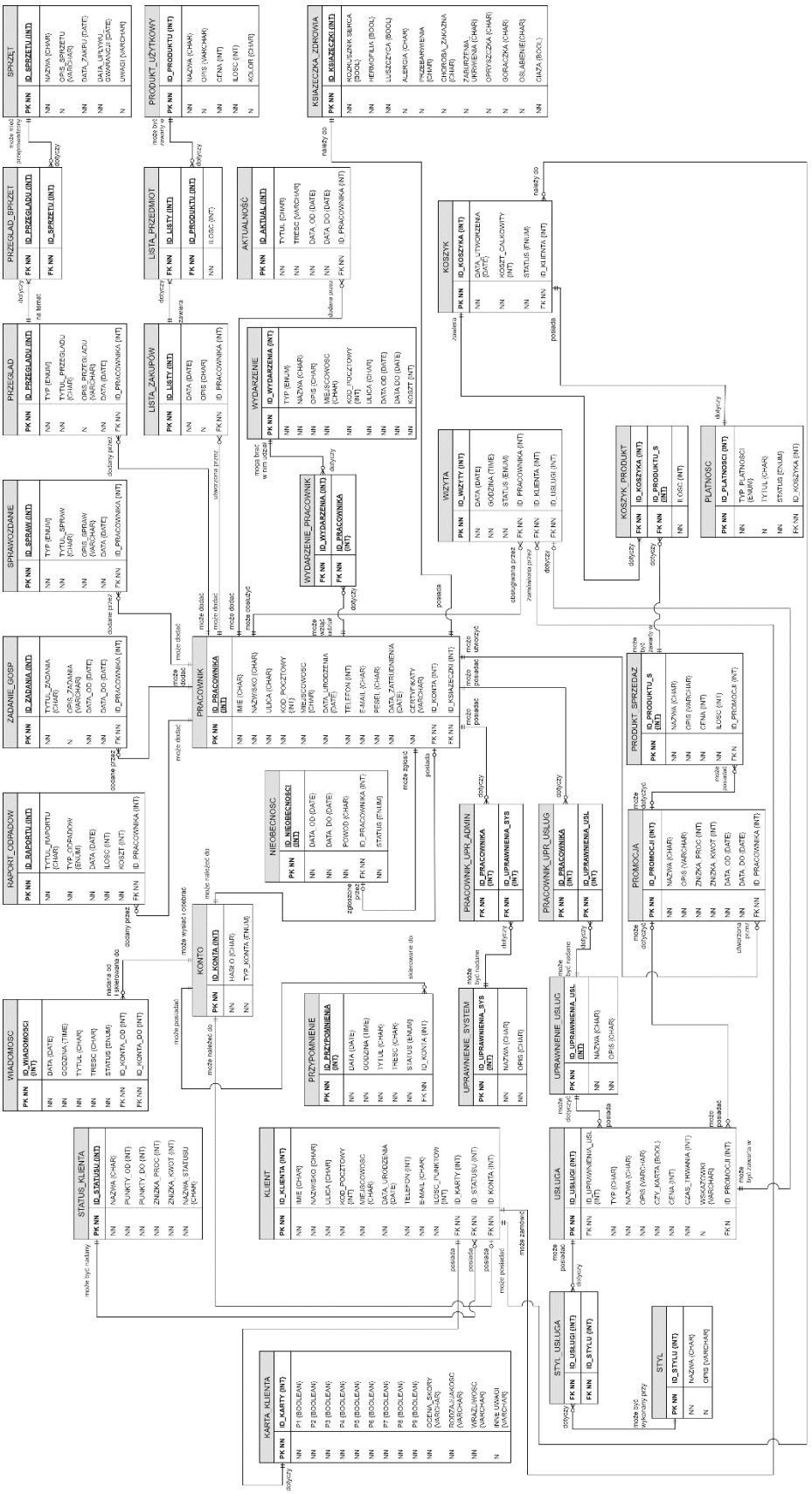
- encja „KONTO" może reprezentować zarówno konto pracownika, klienta jak i administratora. Rozróżniane jest to za pomocą ENUM przyjmujące wartości 'PRACOWNIK', 'ADMINISTRATOR' bądź 'KLIENT'.

- w encji „RAPORT_ODPADÓW" umieszczenie ENUM „TYP_ODPADÓW" umożliwił na scalenie encji reprezentujące różne typy odpadów: 150110 (opakowania, farby, lakiery w pojemnikach szklanych, opakowania zawierające pozostałości substancji niebezpiecznych), 150107 (zużyte opakowania szklane), 150102 (opakowania z tworzyw sztucznych - plastik), 160214 (zużyte urządzenia) oraz 80103 (odpady zakaźne - medyczne zawierające żywe kultury bakterii).

- dla związków M:M (wielu do wielu) utworzone zostały dodatkowe encje przechowywujące klucze główne encji zawierające taki związek pomiędzy sobą, na przykład encja „WYDARZENIE_PRACOWNIK" przechowywująca klucz obcy encji „PRACOWNIK" oraz klucz obcy encji „WYDARZENIE" dzięki czemu wiemy, który pracownik zapisał się na które wydarzenie (np.: pracownica „Agata" będzie mogła wziąć udział w szkoleniu odbywającym się w Poznaniu oraz w innym szkoleniu odbywającym się w Warszawie, a szkolenia te będą mogły posiadać wielu uczestników) bądź na przykład encja „KOSZYK_PRODUKT" informująca nas o tym, w jakiej ilości dany koszyk posiada dane przedmioty (oraz w jakiej ilości dane przedmioty należą do jakich koszyków).

W ostatnim kroku baza została znormalizowana do postaci 2NF, gdzie wymagało to częściowego funkcyjnego uzależnienia kolumn niekluczowych od wszystkich kluczy potencjalnych.

10.2 Schemat ERD po przeprowadzeniu normalizacji



10.3 Skrypt bazy danych

Poniżej przedstawiona została część skryptu odpowiedzialna za utworzenie w języku SQL bazy danych aplikacji wspomagającej pracę gabinetu kosmetycznego. Pełny skrypt wraz z skryptem tworzącym klucze obce znajduje się w oddzielnym pliku.

Zauważyć należy jedynie, iż nie ma możliwość utworzenia zmiennej typu BOOLEAN w systemie MySQL, dlatego wszystkie te wartości zastąpione zostały zmienną typu SHORTINT o długości 1, która przyjmuje wartości odpowiednio 1 bądź 0 dla wartości TRUE oraz FALSE.

We wszystkich tabelach zastosowane zostało również polecenie:
`DEFAULT characterSET utf8mb4 COLLATE utf8mb4_unicode_ci;`
umożliwiające obsługę polskich znaków w bazie danych jak i podczas wymiany danych między bazą danych a aplikacją.

10.3.1 Tabela klient

```
CREATE TABLE klient
 (
    id_klienta      INT(8) PRIMARY KEY auto_increment,
    imie            CHAR(255) NOT NULL,
    nazwisko        CHAR(255) NOT NULL,
    ulica           CHAR(255) NOT NULL,
    kod_pocztowy    CHAR(10) NOT NULL,
    miejscowosc     CHAR(255) NOT NULL,
    data_urodzenia DATE NOT NULL,
    telefon         INT(8) NOT NULL,
    e_mail          CHAR(255) NOT NULL,
    ilosc_punktow   INT(8) NOT NULL,
    id_karty        INT(8) NOT NULL,
    id_statusu      INT(8) NOT NULL,
    id_konta        INT(8) NOT NULL
 )
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

10.3.2 Tabela pracownik

```
CREATE TABLE pracownik
  (
    id_pracownika      INT(8) PRIMARY KEY auto_increment,
    imie               CHAR(255) NOT NULL,
    nazwisko           CHAR(255) NOT NULL,
    ulica              CHAR(255) NOT NULL,
    kod_pocztowy       CHAR(10) NOT NULL,
    miejscowowsc       CHAR(255) NOT NULL,
    data_urodzenia     DATE NOT NULL,
    telefon            INT(8) NOT NULL,
    e_mail             CHAR(255) NOT NULL,
    pesel              INT(8) NOT NULL,
    data_zatrudnienia DATE NOT NULL,
    certyfikaty        VARCHAR(5000) NULL,
    id_konta           INT(8) NOT NULL,
    id_ksiazeczki      INT(8) NOT NULL
  )
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

10.3.3 Tabela konto

```
CREATE TABLE konto
  (
     id_konta  INT(8) PRIMARY KEY auto_increment,
     haslo     CHAR(255) NOT NULL,
     typ_konta ENUM('PRACOWNIK', 'ADMINISTRATOR', 'KLIENT')
  )
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

10.3.4 Tabela wizyta

```
CREATE TABLE wizyta
  (
     id_wizyty     INT(8) PRIMARY KEY auto_increment,
     data          DATE NOT NULL,
     godzina       TIME NOT NULL,
     status        ENUM('DO_ZATWIERDZENIA', 'ODRZUCONE', 'POTWIERDZONE',
     'OPLACONE') NOT
     NULL,
     id_pracownika INT(8) NOT NULL,
     id_klienta    INT(8) NOT NULL,
     id_uslugi     INT(8) NOT NULL
  )
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

10.3.5 Tabela raport_odpadów

```
CREATE TABLE raport_odpadow
  (
     id_raportu    INT(8) PRIMARY KEY auto_increment,
     tytul_raportu CHAR(255) NOT NULL,
     typ_odpadow   ENUM('150110', '150107', '150102', '160214', '180103') NOT
     NULL,
     data          DATE NOT NULL,
     ilos          INT(8) NOT NULL,
     koszt         INT(8) NOT NULL,
     id_pracownika INT(8) NOT NULL
  )
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

10.3.6 Tabela aktualność

```
CREATE TABLE aktualnosc
  (
     id_aktualnosci INT(8) PRIMARY KEY auto_increment,
     tytul          CHAR(255) NOT NULL,
     tresc          VARCHAR(5000) NOT NULL,
     data_od        DATE NOT NULL,
     data_do        DATE NOT NULL,
     id_pracownika  INT(8) NOT NULL
  )
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

10.3.7 Tabela wydarzenie

```
CREATE TABLE wydarzenie
  (
     id_wydarzenia INT(8) PRIMARY KEY auto_increment,
     typ           ENUM('KONGRES', 'SZKOLENIE', 'TARGI_KOSMETYCZNE') NOT NULL,
     nazwa         CHAR(255) NOT NULL,
     opis          CHAR(255) NOT NULL,
     ulica         CHAR(255) NOT NULL,
     kod_pocztowy  CHAR(10) NOT NULL,
     miejscowowsc  CHAR(255) NOT NULL,
     data_od       DATE NOT NULL,
     data_do       DATE NOT NULL,
     kosz          INT(8) NOT NULL
  )
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

10.3.8 Tabela usługa

```
CREATE TABLE usluga
  (
     id_uslugi          INT(8) PRIMARY KEY auto_increment,
     id_uprawnienia_usl INT(8) NOT NULL,
     typ_uslugi         CHAR(255) NOT NULL,
     nazwa              CHAR(255) NOT NULL,
     opis               VARCHAR(5000) NOT NULL,
     czy_karta          BOOLEAN NOT NULL,
     cena               INT(8) NOT NULL,
     czas_trwania       INT(8) NOT NULL,
     wskazowki          VARCHAR(5000) NULL,
     id_promocji        INT(8) NULL
  )
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

10.3.9 Tabela usługa

```
CREATE TABLE promocja
  (
     id_promocji   INT(8) PRIMARY KEY auto_increment,
     nazwa         CHAR(255) NOT NULL,
     opis          VARCHAR(5000) NOT NULL,
     znizka_proc   INT(8) NULL,
     znizka_kwot   INT(8) NULL,
     data_od       DATE NOT NULL,
     data_do       DATE NOT NULL,
     id_pracownika INT(8) NOT NULL
  )
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

10.4 Procedury utworzone dla bazy danych

Poniżej przedstawiona została część skryptu odpowiedzialna za utworzenie w języku SQL procedur wykorzystywanych przez naszą aplikację. W chwili obecnej liczba utworzonych procedur osiągnęła liczbę 42, dlatego pełny skrypt procedur znajduje się w oddzielnym pliku.

10.4.1 Procedura tworząca konto klienta

```
DELIMITER //
CREATE PROCEDURE utworz_klienta(
    p_email         CHAR(255),
    p_haslo         CHAR(255),
    p_imie          CHAR(255),
    p_nazwisko      CHAR(255),
    p_ulica         CHAR(255),
    p_kod_pocztowy  INT(8),
    p_miejscowosc  CHAR(255),
    p_data_urodzenia DATE,
    p_telefon       INT(8))
BEGIN
    IF ((SELECT e_mail FROM klient WHERE e_mail = p_email) IS NULL) THEN
        INSERT INTO konto(haslo, typ_konta) VALUES (p_haslo, 'KLIENT');

        SET @id_konta := (SELECT MAX(id_konta) FROM konto WHERE haslo =
p_haslo);

        INSERT INTO klient(imie, nazwisko, ulica, kod_pocztowy, miejscowosc,
data_urodzenia,telefon, e_mail, ilosc_punktow, id_karty, id_statusu, id_konta)
VALUES (p_imie, p_nazwisko, p_ulica, p_kod_pocztowy, p_miejscowosc,
p_data_urodzenia, p_telefon, p_email, 0, 0, 1, @id_konta);
    END IF;
END //
```

10.4.2 Procedura tworząca konto pracownika

```
DELIMITER //
CREATE PROCEDURE utworz_pracownika(
     p_haslo             CHAR(255),
     p_imie              CHAR(255),
     p_nazwisko          CHAR(255),
     p_ulica             CHAR(255),
     p_kod_pocztowy      INT(8),
     p_miejscowowsc      CHAR(255),
     p_data_urodzenia    DATE,
     p_telefon           INT(8),
     p_e_mail            CHAR(255),
     p_pesel             INT(8),
     p_data_zatrudnienia DATE,
     p_certyfikaty       VARCHAR(5000),
     p_id_konta          INT(8),
     p_rozrusznik_serca     BOOLEAN,
     p_hermofilia           BOOLEAN,
     p_luszczyca            BOOLEAN,
     p_alergia              CHAR(255),
     p_przebarwienie        CHAR(255),
     p_choroba_zakazna      CHAR(255),
     p_zaburzenia_ukrwienia CHAR(255),
     p_opryszczka           CHAR(255),
     p_goraczka             CHAR(255),
     p_oslabienie           CHAR(255),
     p_ciaza                BOOLEAN
)
BEGIN
    IF ((SELECT e_mail FROM pracownik WHERE e_mail = p_e_mail) IS NULL) THEN
        INSERT INTO konto(haslo, typ_konta) VALUES (p_haslo, 'PRACOWNIK');
        INSERT INTO ksiazeczka_zdrowia(rozrusznik_serca, hermofilia,
luszczyca, alergia, przebarwienie, choroba_zakazna, zaburzenia_ukrwienia,
opryszczka, goraczka, oslabienie, ciaza)
        VALUES ( p_rozrusznik_serca, p_hermofilia, p_luszczyca, p_alergia,
p_przebarwienie, p_choroba_zakazna, p_zaburzenia_ukrwienia, p_opryszczka,
p_goraczka, p_oslabienie, p_ciaza);

        SET @id_konta := (SELECT MAX(id_konta) FROM konto WHERE haslo =
p_haslo);
        SET @id_ksiazeczki := (SELECT MAX(id_ksiazeczki) FROM
ksiazeczka_zdrowia);
```

10.4.3 Procedura tworząca usługę

```
DELIMITER //
CREATE PROCEDURE dodaj_usluge(
    P_ID_UPRAWNIENIA_USL INT(8),
    P_TYP_USLUGI CHAR(255),
    P_NAZWA CHAR(255),
    P_OPIS VARCHAR(5000),
    P_CZY_KARTA BOOLEAN,
    P_CENA INT(8),
    P_CZAS_TRWANIA INT(8),
    P_WSKAZOWKI VARCHAR(5000)
)
BEGIN
    IF ((SELECT MAX(ID_USLUGI) FROM usluga) IS NULL) THEN
        SET @ID_USLUGI := 1;
    ELSE
        SET @ID_USLUGI := (SELECT MAX(ID_USLUGI) FROM usluga) + 1;
    END IF;
    INSERT INTO usluga(ID_USLUGI, ID_UPRAWNIENIA_USL, TYP_USLUGI, NAZWA, OPIS,
CZY_KARTA, CENA, CZAS_TRWANIA, WSKAZOWKI) VALUES (@ID_USLUGI,
P_ID_UPRAWNIENIA_USL, P_TYP_USLUGI, P_NAZWA, P_OPIS, P_CZY_KARTA, P_CENA,
P_CZAS_TRWANIA, P_WSKAZOWKI);
END //
```

10.4.4 Procedura tworząca wydarzenie (szkolenie / kongres / targi)

```
DELIMITER //
CREATE PROCEDURE dodaj_wydarzenie(
    P_TYP ENUM('KONGRES', 'SZKOLENIE', 'TARGI_KOSMETYCZNE'),
    P_NAZWA CHAR(255),
    P_OPIS CHAR(255),
    P_MIEJSCOWOSC CHAR(255),
    P_KOD_POCZTOWY INT(5),
    P_ULICA CHAR(255),
    P_DATA_OD DATE,
    P_DATA_DO DATE,
    P_KOSZ INT(8),
    P_ID_PRACOWNIKA INT(8)
)
BEGIN
    INSERT INTO wydarzenie(TYP, NAZWA, OPIS, MIEJSCOWOSC, KOD_POCZTOWY, ULICA,
DATA_OD, DATA_DO, KOSZ) VALUES (P_TYP, P_NAZWA, P_OPIS, P_MIEJSCOWOSC,
P_KOD_POCZTOWY, P_ULICA, P_DATA_OD, P_DATA_DO, P_KOSZ);
    SET @ID_WYDARZENIA := (SELECT ID_WYDARZENIA FROM wydarzenie WHERE NAZWA =
P_NAZWA);
    INSERT INTO wydarzenie_pracownik(ID_WYDARZENIA, ID_PRACOWNIKA) VALUES
(@ID_WYDARZENIA, P_ID_PRACOWNIKA);
END //
```

10.4.5 Procedura tworząca przegląd sprzętu

```
DELIMITER //
CREATE PROCEDURE przeglad(
    P_TYTUL_PRZEGLADU CHAR(255),
    P_OPIS_PRZEGLADU VARCHAR(5000),
    P_DATA DATE,
    P_ID_PRACOWNIKA INT(8),
    P_ID_SPRZETU INT(8)
)
BEGIN
    INSERT INTO przeglad(TYTUL_PRZEGLADU, OPIS_PRZEGLADU, DATA, ID_PRACOWNIKA)
VALUES (P_TYTUL_PRZEGLADU, P_OPIS_PRZEGLADU, P_DATA, P_ID_PRACOWNIKA);

    SET @ID_PRZEGLADU := (SELECT ID_PRZEGLADU FROM przeglad WHERE ID_PRZEGLADU
NOT IN(SELECT ID_PRZEGLADU FROM PRZEGLAD_SPRZET));

    INSERT INTO przeglad_sprzet(ID_PRZEGLADU, ID_SPRZETU) VALUES
(@ID_PRZEGLADU, P_ID_SPRZETU);
END //
```

10.4.6 Procedura tworząca nieobecność pracownika

```
DELIMITER //
CREATE PROCEDURE zglos_nieobecnosc(
    P_DATA_OD DATE,
    P_DATA_DO DATE,
    P_POWOD CHAR(255),
    P_ID_PRACOWNIKA INT(8),
    P_STATUS ENUM('NIEPOTWIERDZONE', 'POTWIERDZONE')
)
BEGIN
    INSERT INTO nieobecnosc(DATA_OD, DATA_DO, POWOD, ID_PRACOWNIKA, STATUS)
VALUES (P_DATA_OD, P_DATA_DO, P_POWOD, P_ID_PRACOWNIKA, P_STATUS);
END //
```

10.4.7 Procedura tworząca wiadomość od użytkownika do użytkownika

```
DELIMITER //
CREATE PROCEDURE wiadomosc(
    P_DATA DATE,
    P_GODZINA TIME,
    P_TYTUL CHAR(255),
    P_TRESC CHAR(255),
    P_STATUS ENUM('WYSLANA', 'ODCZYTANA'),
    P_ID_KONTA_OD INT(8),
    P_ID_KONTA_DO INT(8)
)
BEGIN
    INSERT INTO wiadomosc(DATA, GODZINA, TYTUL, TRESC, STATUS_WIADOMOSCI,
ID_KONTA_OD, ID_KONTA_DO) VALUES (P_DATA, P_GODZINA, P_TYTUL, P_TRESC,
P_STATUS, P_ID_KONTA_OD, P_ID_KONTA_DO);
END //
```

10.4.8 Procedura dodająca produkt do danej promocji

```
DELIMITER //
CREATE PROCEDURE promocja_produkt(
    P_ID_PROMOCJI INT(8),
    P_ID_PRODUKTU_S INT(8)
)
BEGIN
    UPDATE produkt_sprzedaz SET ID_PROMOCJI = P_ID_PROMOCJI WHERE ID_PRODUKTU
= P_ID_PRODUKTU;
END //
```

10.5 Schemat bazy danych w systemie