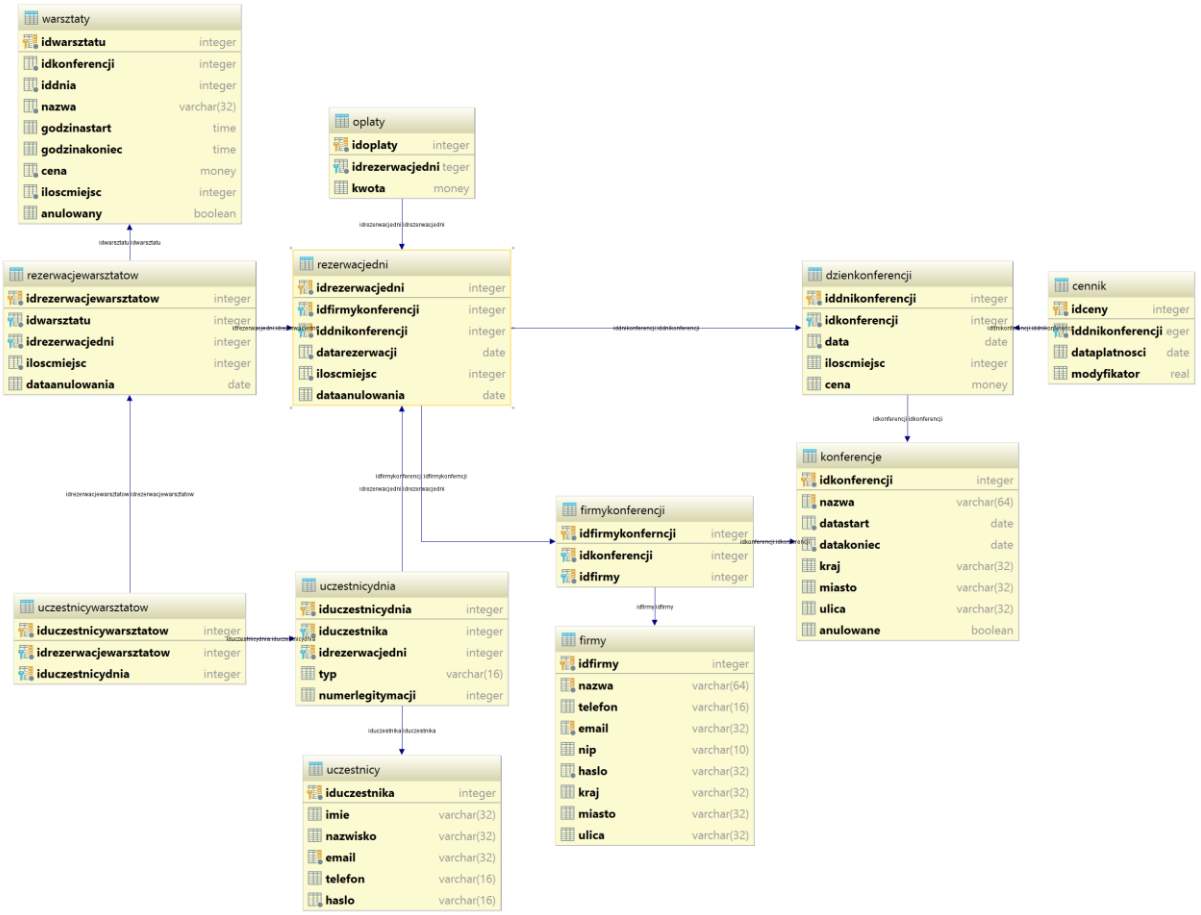
Projekt systemu bazodanowego Konferencje

Projekt miał na celu zaprojektowanie i implementacje systemu obsługi działalności firmy organizującej konferencje. Klienci mogą rejestrować się na konferencje za pomocą systemu WWW. Rezerwacje są na poszczególny dzień. Klientami mogą być zarówno indywidualne osoby jak i firmy, natomiast uczestnikami konferencji są pojedyncze osoby. Firma ma możliwość rezerwacji określonej liczby miejsc bez podania danych uczestników. Dodatkowo uczestnik danego dnia konferencji może zarejestrować się na warsztaty w tym dniu. Wysokość opłaty zależy od zarezerwowanych usług i terminu ich rezerwacji. Projekt został zaimplementowany w oparciu o serwer PostgreSQL.

1. Projekt bazy danych



2.Tworzenie tabel

2.1 Tabela Firmy

Tabela zawiera informacje o firmach, jeden rekord odpowiada jednej firmie. Konta w systemie WWW, wykorzystują email jako login i hasło. Dodatkowo przechowywane są dane adresowe niezbędne do wystawienia faktury.

```
CREATE TABLE public.firmy
(
    idfirmy SERIAL PRIMARY KEY NOT NULL,
    nazwa VARCHAR(64) UNIQUE NOT NULL,
    telefon VARCHAR(16) CHECK (telefon NOT LIKE '%[^0-9]%'),
    email VARCHAR(32) UNIQUE NOT NULL ,
    nip VARCHAR(10) CHECK (nip NOT LIKE '%[^0-9]%'),
    haslo VARCHAR(32) NOT NULL,
    kraj VARCHAR(32) CHECK (kraj !~ '[^A-Z a-zążółćśęźńĄŻÓŁĆŚĘŹŃ-]'),
    miasto VARCHAR(32) CHECK (miasto !~ '[^A-Z a-zążółćśęźńĄŻÓŁĆŚĘŹŃ-]'),
    ulica VARCHAR(32)
);
```

2.2 Tabela Konferencje

Tabela zawiera szczegółowe informacje o każdej organizowanej konferencji, takie jak nazwa, lokalizacja wydarzenia oraz daty rozpoczęcia i zakończenia. Tabela posiada również pole ‘anulowane’ co umożliwia proste odwołanie konferencji w systemie.

```
CREATE TABLE public.konferencje
(
    idkonferencji SERIAL PRIMARY KEY NOT NULL,
    nazwa VARCHAR(64) UNIQUE NOT NULL,
    datastart DATE NOT NULL,
    datakoniec DATE NOT NULL CHECK (konferencje.datakoniec >=
konferencje.datastart),
    kraj VARCHAR(32) CHECK (kraj !~ '[^A-Z a-zążółćśęźńĄŻÓŁĆŚĘŹŃ-]'),
    miasto VARCHAR(32) CHECK (miasto !~ '[^A-Z a-zążółćśęźńĄŻÓŁĆŚĘŹŃ-]'),
    ulica VARCHAR(32),
    anulowane BOOLEAN
);
```

2.3 Tabela Warsztaty

Tabela zawiera dane o warsztatach, najważniejsze z nich to ilość miejsc na warsztacie i godziny rozpoczęcia i zakończenia.

```
CREATE TABLE public.warsztaty
(
    idwarsztatu SERIAL PRIMARY KEY NOT NULL,
    idkonferencji INTEGER NOT NULL,
    iddnia INTEGER NOT NULL,
    nazwa VARCHAR(32) NOT NULL,
    godzinastart TIME,
    godzinakoniec TIME CHECK (warsztaty.godzinastart <
warsztaty.godzinakoniec),
    cena MONEY NOT NULL CHECK (warsztaty.cena >= 0.00::MONEY),
    iloscmiejsc INTEGER NOT NULL CHECK (warsztaty.iloscmiejsc > 0),
    anulowany BOOLEAN
);
```

2.4 Tabela Uczestnicy

Tabela zawiera dane osobowe o uczestnikach. Do logowania do systemu WWW jest używany email i hasło.

```
CREATE TABLE uczestnicy
(
    iduczestnika SERIAL PRIMARY KEY NOT NULL,
    imie VARCHAR(32),
    nazwisko VARCHAR(32),
    email VARCHAR(32) UNIQUE NOT NULL ,
    telefon VARCHAR(16) CHECK (telefon NOT LIKE '%[^0-9]%'),
    haslo VARCHAR(16) NOT NULL
);
```

2.5 Tabela Dzień Konfernecji

Tabela zawiera dane o poszczególnych dniach konferencji, takie jak ilość miejsc, bazowa cena oraz data.

```
CREATE  TABLE public.dzienkonferencji(
  iddnikonferencji SERIAL NOT NULL PRIMARY KEY,
  idkonferencji INTEGER NOT NULL REFERENCES konferencje (idkonferencji),
  data DATE NOT NULL,
  iloscmiejsc INTEGER CHECK (dzienkonferencji.iloscmiejsc >= 0),
  cena MONEY CHECK (dzienkonferencji.cena>=0.00::MONEY)
);
```

2.6 Tabela Firmy Konferencji

Tabela łącznikowa między tabelami Konferencje i Firmy.

```
CREATE TABLE public.firmykonferencji (
  idfirmykonferncji SERIAL NOT NULL PRIMARY KEY ,
  idkonferencji INTEGER NOT NULL REFERENCES konferencje (idkonferencji),
  idfirmy INTEGER NOT NULL REFERENCES firmy (idfirmy),
  CONSTRAINT no_duplicate UNIQUE (idkonferencji,idfirmy)
);
```

2.7 Tabela Rezerwacje Dni

Tabela zawiera dane o rezerwacjach klientów na poszczególne dni. Posiada pole data anulowania, które pozwala anulować rezerwację.

```
CREATE TABLE public.rezerwacjedni (
  idrezerwacjedni SERIAL NOT NULL PRIMARY KEY,
  idfirmykonferencji INTEGER NOT NULL REFERENCES firmykonferencji
(idfirmykonferncji),
  iddnikonferencji INTEGER NOT NULL REFERENCES dzienkonferencji
(iddnikonferencji),
  datarezerwacji DATE NOT NULL,
  iloscmiejsc INTEGER NOT NULL,
  dataanulowania DATE,
);
```

2.8 Tabela Uczestnicy Dnia

Tabela zawiera dane o uczestnikach, którzy po rezerwacji opłacili swój udział w danym dniu konferencji, przechowujemy tu również numer legitymacji studenckiej, jeżeli klientowi przysługuje zniżka.

```
CREATE TABLE public.uczestnicydnia (
  iduczestnicydnia SERIAL NOT NULL PRIMARY KEY,
  iduczestnika INTEGER NOT NULL REFERENCES uczestnicy (iduczestnika),
  idrezerwacjedni INTEGER NOT NULL REFERENCES rezerwacjedni
(idrezerwacjedni),
  typ VARCHAR(16),
  numerlegitymacji INTEGER CHECK (uczestnicydnia.numerlegitymacji >= 0),
  CONSTRAINT uczestnicydnia_no_duplicate UNIQUE (iduczestnika,
idrezerwacjedni)
);
```

2.9 Tabela Rezerwacje Warsztatów

Tabela zawiera dane o rezerwacjach warsztatów. Wpis w tej tabeli wymaga wpisu w tabeli rezerwacje dnia.

```
CREATE TABLE public.rezerwacjewarsztatow (
  idrezerwacjewarsztatow SERIAL NOT NULL PRIMARY KEY,
  idwarsztatu INTEGER NOT NULL REFERENCES warsztaty (idwarsztatu),
  idrezerwacjedni INTEGER NOT NULL REFERENCES rezerwacjedni
(idrezerwacjedni),
  iloscmiejsc INTEGER NOT NULL CHECK (rezerwacjewarsztatow.iloscmiejsc >
0),
  dataanulowania DATE
);
```

2.10 Tabela Opłaty

Tabela zawiera dane o warsztatach

```
CREATE TABLE public.oplaty (
  idoplaty SERIAL NOT NULL PRIMARY KEY,
  idrezerwacjedni INTEGER NOT NULL REFERENCES rezerwacjedni
(idrezerwacjedni),
  kwota MONEY CHECK (kwota >= 0.00::MONEY)
);
```

2.11 Tabela Cennik

Tabela zawiera dane o warsztatach

```
CREATE TABLE public.cennik (
  idceny SERIAL NOT NULL PRIMARY KEY,
  iddnikonferencji INTEGER NOT NULL REFERENCES dzienkonferencji
(iddnikonferencji),
  dataplatnosci DATE,
  modyfikator REAL
);
```

2.12 Tabela Uczestnicy Warsztatów

Tabela zawiera dane o warsztatach

```
CREATE TABLE uczestnicywarsztatow
(
    iduczestnicywarsztatow SERIAL PRIMARY KEY NOT NULL,
    idrezerwacjewarsztatow INTEGER NOT NULL REFERENCES rezerwacjewarsztatow
(idrezerwacjewarsztatow),
    iduczestnicydnia INTEGER NOT NULL  REFERENCES uczestnicydnia
(iduczestnicydnia),
    CONSTRAINT uczestnicywarsztatow_no_duplicate UNIQUE
(idrezerwacjewarsztatow, iduczestnicydnia)
);
```

3.Widoki

3.1 Widok zaplanowane konferencje – pokazuje wszystkie konferencje, które jeszcze się nie odbyły.

```
CREATE VIEW public.ZaplanowaneKonferencje AS
  SELECT idkonferencji, nazwa, datastart FROM konferencje
  WHERE konferencje.datastart > CURRENT_DATE;
```

3.2 Widok nieopłacone rezerwacje – tworzy tabelę IdRezerwacji, które jeszcze nie zostały opłacone, ani anulowane.

```
CREATE VIEW public.NiepolaconeRezerwacjeKonferencji AS
  SELECT rezerwacjedni.idrezerwacjedni AS "Id Rezerwacji" FROM
rezerwacjedni
  LEFT OUTER JOIN oplaty ON rezerwacjedni.idrezerwacjedni =
oplaty.idrezerwacjedni
  WHERE (idoplaty IS NULL) OR (dataanulowania IS NOT NULL);
```

3.3 Widok zaplanowane warsztaty – tworzy tabelę warsztatów, które jeszcze się nie odbyły oraz ilość pozostałych wolnych miejsc na nie.

```
CREATE VIEW public.ZaplanowaneWarsztaty AS
  SELECT warsztaty.nazwa, warsztaty.iddnia, warsztaty.godzinakoniec,
warsztaty.cena,
    warsztaty.iloscmiejsc -(SELECT SUM(rezerwacjewarsztatow.iloscmiejsc)
FROM rezerwacjewarsztatow
                            WHERE (rezerwacjewarsztatow.idwarsztatu =
warsztaty.idwarsztatu)
                            AND rezerwacjewarsztatow.dataanulowania IS
NULL) AS "Dostępne miejsca"
FROM warsztaty
INNER JOIN dzienkonferencji ON warsztaty.iddnia =
dzienkonferencji.iddnikonferencji
WHERE dzienkonferencji.data > CURRENT_DATE;
```

3.4 Widok aktywność uczestników – tworzy tabelę uczestników i ilość dni konferencji, w których wzięli oni udział.

```
CREATE VIEW public.AktywnoscUczestnikow AS
  SELECT iduczestnika, SUM(iduczestnicydnia) AS "Aktywność" FROM
uczestnicydnia
  GROUP BY iduczestnika
  ORDER BY SUM(iduczestnicydnia) DESC;
```

3.5 Widok aktywność firm – tworzy tabelę firm i ilość rezerwacji na dni konferencji.

```
CREATE VIEW public.AktywnoscFirm AS
  SELECT idfirmy, COUNT(idfirmykonferncji) AS "Aktywność" FROM
firmykonferencji
  INNER JOIN rezerwacjedni ON firmykonferencji.idfirmykonferncji =
rezerwacjedni.idfirmykonferencji
  GROUP BY idfirmy, dataanulowania
  HAVING (dataanulowania IS NULL)
  ORDER BY COUNT(idfirmykonferncji) DESC;
```

4. Funkcje pomocnicze dla constraintów i trigerrów

4.1 Funkcje użyte w trigerrach – funkcja sprawdza czy godzinna rozpoczęcia jest przed godziną zakończenia. W przeciwnym przypadku cofa transakcję.

```
CREATE OR REPLACE FUNCTION public.sprawdz_godziny_warsztatu()
 RETURNS trigger
 LANGUAGE plpgsql
AS $function$
  BEGIN
    IF NEW.godzinastart > NEW.godzinakoniec THEN
      RAISE 'Niepoprawne godziny';
      ROLLBACK TRANSACTION;
    ELSE
      RETURN NEW;
    END IF;
  END;
  $function$;
```

Funkcja anuluje rezerwacje warsztatów, odpowiadających anulowanym rezerwacjom dni.

```
CREATE or REPLACE FUNCTION public.anuluj_rezerwacje_warsztatu () RETURNS
TRIGGER
 LANGUAGE plpgsql
AS $$
  BEGIN
    IF NEW.dataanulowania IS NOT NULL THEN
      UPDATE rezerwacjewarsztatow SET dataanulowania = CURRENT_DATE
      WHERE rezerwacjewarsztatow.idrezerwacjedni = OLD.idrezerwacjedni;
    ELSE
      RETURN OLD;
    END IF;
  END;
$$;
```

4.2 Pozostałe Funkcje – dodające wpisy do tabel

```
CREATE OR REPLACE FUNCTION Dodaj_Cennik(
  Konferencja character varying,
  Dzien date,
  DataPlatnosci date,
  Cena MONEY
) returns void as $$
  DECLARE Id_Konferencji INTEGER;
  DECLARE Id_Dnia_Konferencji INTEGER;
  BEGIN
    INSERT INTO cennik
    VALUES(DEFAULT, Znajdz_Dzien_Konferencji(Konferencja, Dzien),
DataPlatnosci, Cena);
  END;
  $$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Dodaj_Dzien_Konferencji(
  Konferencja character varying,
  Data date,
  Cena MONEY,
  miejsca integer
) returns void as $$
  BEGIN
    INSERT INTO dzienkonferencji VALUES(DEFAULT,
Znajdz_Konferencje(Konferencja), Data, Cena, Miejsca);
  END;
  $$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Dodaj_Firme(
  Firma character varying,
  Telefon character varying,
  Email character varying,
  NIP character varying,
  Haslo character varying,
  Kraj character varying,
  Miasto character varying,
  Ulica character varying
) returns void as $$
  BEGIN
    INSERT INTO firmy VALUES(DEFAULT, Firma, Telefon, Email, NIP, Haslo,
Kraj, Miasto, Ulica);
  END
  $$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Dodaj_Firme_Konferencji(
  Konferencja character varying,
  Firma character varying
  ) returns void as $$
  DECLARE Id_Konferencji INTEGER;
  DECLARE Id_Firmy INTEGER;
  BEGIN
    ID_Konferencji := Znajdz_Konferencje(Konferencja);
    Id_Firmy := Znajdz_Firme(Firma);
    INSERT INTO firmykonferencji
    VALUES(DEFAULT, Id_Konferencji, Id_Firmy);
  END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Dodaj_Konferencje(
  Konferencja character varying,
  datastart date,
  datakoniec date,
  kraj character varying,
  miasto character varying,
  ulica character varying
  )
 RETURNS void
 LANGUAGE plpgsql
AS $function$
BEGIN
  INSERT INTO konferencje VALUES(DEFAULT, Konferencja, DataStart,
DataKoniec, Kraj, Miasto, Ulica, false);
END; $function$;
```

```
CREATE OR REPLACE FUNCTION Dodaj_Oplate(
  Firma       character varying,
  Konferencja character varying,
  Dzien       date,
  Kwota       money
) RETURNS VOID AS $$
  DECLARE Id_Rezerwacji_Dnia INTEGER;
  BEGIN
    Id_Rezerwacji_Dnia=Znajdz_Rezerwacje_Dnia(Firma, Konferencja, Dzien);
    INSERT INTO oplaty
    VALUES(DEFAULT, Id_Rezerwacji_Dnia, Kwota);
  END;
  $$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Dodaj_Rezerwacje_Dni(
  Firma character varying,
  Konferencja character varying,
  Dzien date,
  data_rezerwacji date,
  IloscMiejsc integer
) returns void as $$
  DECLARE Id_Dnia_Konferencji Integer;
  DECLARE Id_Firmy_Konferencji Integer;
  BEGIN
    ID_Dnia_Konferencji := Znajdz_Dzien_Konferencji(Konferencja, Dzien);
    Id_Firmy_Konferencji := Znajdz_Firme_Konferencji(Firma, Konferencja);
    INSERT INTO rezerwacjedni
    VALUES(DEFAULT, Id_Firmy_Konferencji, Id_Dnia_Konferencji,
data_rezerwacji, IloscMiejsc, NULL);
  END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Dodaj_Rezerwacje_Warsztatu(
  Warsztat character varying,
  Konferencja character varying,
  Dzien date,
  Firma character varying,
  Ilosc_Miejsc INTEGER
) returns void as $$
  BEGIN
    INSERT INTO rezerwacjewarsztatow
    VALUES(DEFAULT, Znajdz_Warsztat(Warsztat, Konferencja, Dzien),
      Znajdz_Rezerwacje_Dnia(Firma, Konferencja, Dzien), Ilosc_Miejsc,
NULL);
  END;
  $$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Dodaj_Uczestnika(
    Imie character varying,
    Nazwisko character varying,
    Email character varying,
    Telefon character varying,
    Haslo character varying
) returns void as $$
  BEGIN
    INSERT INTO uczestnicy VALUES(DEFAULT, Imie, Nazwisko, Email, Telefon,
Haslo);
  END;
$$ LANGUAGE plpgsql;;
```

```
CREATE FUNCTION dodaj_uczestnika_dnia (user_email character varying,
konferencja character varying, dzien date, firma character varying, typ
character varying, numer_legitymacji integer) RETURNS void
 LANGUAGE plpgsql
AS $$
  DECLARE Id_Rezerwacji_Dnia  INTEGER;
  DECLARE ID_Konferencji      INTEGER;
  DECLARE Id_Dnia_Konferencji INTEGER;
  DECLARE Id_Uczestnika INTEGER;
  BEGIN
    Id_Dnia_Konferencji := Znajdz_Dzien_Konferencji(Konferencja, Dzien);
    Id_Rezerwacji_Dnia := Znajdz_Rezerwacje_Dnia(Firma, Konferencja,
Dzien);
    Id_Uczestnika := znajdz_uczestnika(user_email);
    INSERT INTO uczestnicydnia
    VALUES(DEFAULT, Id_Uczestnika, Id_Rezerwacji_Dnia, Typ,
Numer_Legitymacji);
  END;

$$
```

```
CREATE OR REPLACE FUNCTION Dodaj_Uczestnika_Warsztatu(
    Firma          character varying,
    Warsztat       character varying,
    Konferencja    character varying,
    Dzien          date,
    Id_Uczestnika Integer

    ) RETURNS VOID AS $$
    DECLARE Id_Rezerwacji_Warsztatu INTEGER;
    DECLARE Id_Uczestnika_Dnia INTEGER;
    BEGIN
    Id_Rezerwacji_Warsztatu := Znajdz_Rezerwacje_Warsztatu(Firma, Warsztat,
Konferencja, Dzien);
    Id_Uczestnika_Dnia      := Znajdz_Uczestnika_Dnia(Id_Uczestnika,
Konferencja, Dzien, Firma);
    INSERT INTO uczestnicywarsztatow
    VALUES(DEFAULT, Id_Rezerwacji_Warsztatu, Id_Uczestnika_Dnia);
    END;
    $$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Dodaj_Warsztat(
  Warsztat       character varying,
  Konferencja    character varying,
  Dzien          date,
  Godzinastart   TIME,
  Godzinakoniec TIME,
  Cena           MONEY,
  IloscMiejsc    integer
) returns void as $$
  DECLARE Id_Konferencji      INTEGER;
  DECLARE Id_Dnia_Konferencji INTEGER;
  BEGIN
  INSERT INTO warsztaty
  VALUES(DEFAULT, Znajdz_Dzien_Konferencji(Konferencja, Dzien),
    Godzinastart, Godzinakoniec, Cena, IloscMiejsc, FALSE, Warsztat);
  END;
  $$ LANGUAGE plpgsql;
```

4.3 Funkcje pomocnicze do funkcji dodających

```
CREATE OR REPLACE FUNCTION Znajdz_Konferencje(Konferencja character
varying)
returns Integer as $$
  DECLARE ID_Konferencji INTEGER;
  BEGIN
    ID_Konferencji := (
      SELECT idkonferencji FROM konferencje WHERE
Konferencja=konferencje.nazwa
    );
    IF Id_Konferencji IS NULL THEN
      RAISE 'Nie ma takiej konferencji';
    ELSE return Id_Konferencji;
    END IF;
  END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Znajdz_Dzien_Konferencji(Konferencja character
varying, Dzien date)
returns Integer as $$
  DECLARE Id_Dnia_Konferencji INTEGER;
  BEGIN
    ID_Dnia_Konferencji := (
      SELECT iddnikonferencji FROM dzienkonferencji WHERE
dzienkonferencji.data = dzien AND idkonferencji =
Znajdz_Konferencje(Konferencja)
    );
    IF Id_Dnia_Konferencji IS NULL THEN
      RAISE 'Nie ma takiego dnia konferencji';
    ELSE return Id_Dnia_Konferencji;
    END IF;
  END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION  Znajdz_Firme(Firma character varying)
returns Integer as $$
  DECLARE Id_Firmy INTEGER;
  BEGIN
    IF Firma IS NULL THEN
      Id_Firmy := (SELECT idfirmy FROM firmy WHERE Nazwa='INDIVIDUAL');
    ELSE
      Id_Firmy := (SELECT idfirmy FROM firmy WHERE Nazwa=firma       );
    END IF;
    IF Id_Firmy IS NULL THEN
      RAISE 'Nie ma takiej firmy';
    ELSE return Id_Firmy;
    END IF;
  END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Znajdz_Firme_Konferencji(Firma character
varying, Konferencja character varying)
returns Integer as $$
  DECLARE Id_Firmy_Konferencji Integer;
  BEGIN
    Id_Firmy_Konferencji := (
      SELECT idfirmykonferncji FROM firmykonferencji
      WHERE idfirmy=Znajdz_Firme(Firma) AND
idkonferencji=Znajdz_Konferencje(Konferencja)
      );
    IF Id_Firmy_Konferencji IS NULL THEN
      RAISE 'Ta firma nie jest zarejestrowana na tej konferencji';
    ELSE
      return Id_Firmy_Konferencji;
    END IF;
  END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Znajdz_Rezerwacje_Dnia(
  Firma       character varying,
  Konferencja character varying,
  Dzien       date
)
  returns   Integer as $$
  DECLARE Id_Rezerwacji_Dnia Integer;
  DECLARE Id_Dnia_Konferencji Integer;
  DECLARE Id_Firmy_Konferencji Integer;
  BEGIN
    Id_Dnia_Konferencji    := Znajdz_Dzien_Konferencji(Konferencja, Dzien);
    Id_Firmy_Konferencji   := Znajdz_Firme_Konferencji(Firma, Konferencja);
    Id_Rezerwacji_Dnia     := (
      SELECT  idrezerwacjedni FROM rezerwacjedni
      WHERE   iddnikonferencji=Id_Dnia_Konferencji
      AND     idfirmykonferencji=Id_Firmy_Konferencji
      );
    IF Id_Rezerwacji_Dnia IS NULL THEN
      RAISE 'Dana firma nie ma rezerwacji na tej konferencji tego dnia';
    ELSE
      return Id_Rezerwacji_Dnia;
    END IF;
  END;
  $$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Znajdz_Rezerwacje_Warsztatu(
  Firma       character varying,
  Warsztat    character varying,
  Konferencja character varying,
  Dzien       date
  )
  returns     INTEGER
  as $$
  DECLARE Id_Rezerwacji_Warsztatu INTEGER;
  BEGIN
    Id_Rezerwacji_Warsztatu := (
        SELECT idrezerwacjewarsztatow FROM rezerwacjewarsztatow
        WHERE idwarsztatu=Znajdz_Warsztat(Warsztat, Konferencja, Dzien)
        AND idrezerwacjedni=Znajdz_Rezerwacje_Dnia(Firma, Konferencja,
Dzien)
      );
    IF Id_Rezerwacji_Warsztatu IS NULL THEN
      RAISE 'Nie ma takiej rezerwacji warsztatu';
    ELSE
      return Id_Rezerwacji_Warsztatu;
    END IF;
  END;
  $$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Znajdz_Uczestnika_Dnia(
  Id_Uczestnika Integer,
  Konferencja   character varying,
  Dzien         date,
  Firma         character varying
) returns       Integer
  as $$
  DECLARE Id_Uczestnika_Dnia Integer;
  BEGIN
    Id_Uczestnika_Dnia := (
      SELECT  iduczestnicydnia FROM uczestnicydnia
      WHERE   iduczestnika=Id_Uczestnika
      AND     idrezerwacjedni=Znajdz_Rezerwacje_Dnia(Firma, Konferencja,
Dzien)
      );
    IF Id_Uczestnika_Dnia IS NULL THEN
      RAISE 'Nie ma takiego uczestnika tego dnia';
    ELSE return Id_Uczestnika_Dnia;
    END IF;
  END;
  $$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION Znajdz_Warsztat(
  Warsztat    character varying,
  Konferencja character varying,
  Dzien       date
) returns     Integer as $$
  DECLARE Id_Warsztatu Integer;
  BEGIN
    Id_Warsztatu := (
      SELECT idwarsztatu FROM warsztaty
      WHERE iddnia=Znajdz_Dzien_Konferencji(Konferencja, Dzien)
      AND warsztaty.nazwa=Warsztat
      );
    IF Id_Warsztatu IS NULL THEN
      RAISE 'Nie ma takiego warsztatu';
    ELSE
      return Id_Warsztatu;
    END IF;
  END;
  $$ LANGUAGE plpgsql;
```

```
CREATE FUNCTION znajdz_uczestnika_dnia (id_uczestnika integer, konferencja
character varying, dzien date, firma character varying) RETURNS integer
 LANGUAGE plpgsql
AS $$
  DECLARE Id_Uczestnika_Dnia Integer;
  BEGIN
    Id_Uczestnika_Dnia := (
      SELECT  iduczestnicydnia FROM uczestnicydnia
      WHERE   iduczestnika=Id_Uczestnika
      AND     idrezerwacjedni=Znajdz_Rezerwacje_Dnia(Firma, Konferencja,
Dzien)
      );
    IF Id_Uczestnika_Dnia IS NULL THEN
      RAISE 'Nie ma takiego uczestnika tego dnia';
    ELSE return Id_Uczestnika_Dnia;
    END IF;
  END;

$$
```

4.4 Funkcja anulująca rezerwację dnia

```
  CREATE FUNCTION anuluj_rezerwacje_dnia (idrezewacjidnia INTEGER) RETURNS
void
  LANGUAGE plpgsql
AS $$
  BEGIN
    UPDATE rezerwacjedni SET dataanulowania = CURRENT_DATE;
  END;
$$;
```

4.5 Funkcje tworzące tabelę uczestników dnia i warsztatów, które można np. użyć do generowania etykiet osobowych

```
CREATE OR REPLACE FUNCTION uczestnicy_dnia_konferencji (iddniakonferencji
INTEGER)
RETURNS TABLE ("id uczestnika" INTEGER, "imie" VARCHAR(50), "nazwisko"
VARCHAR(50))
LANGUAGE plpgsql
AS $$
  BEGIN
  RETURN QUERY
  SELECT uczestnicy.iduczestnika, imie, nazwisko FROM uczestnicy
  INNER JOIN uczestnicydnia ON
uczestnicy.iduczestnika=uczestnicydnia.iduczestnika
  INNER JOIN rezerwacjedni ON uczestnicydnia.idrezerwacjedni =
rezerwacjedni.idrezerwacjedni
  INNER JOIN dzienkonferencji ON rezerwacjedni.iddnikonferencji =
dzienkonferencji.iddniakonferencji
  WHERE dzienkonferencji.iddniakonferencji =
uczestnicy_dnia_konferencji.iddniakonferencji
  ORDER BY uczestnicy.iduczestnika;
  END;
$$;
```

```
CREATE OR REPLACE FUNCTION uczestnicy_warsztatu (idwarsztatu INTEGER)
RETURNS TABLE ("id uczestnika" INTEGER, "imie" VARCHAR(50), "nazwisko"
VARCHAR(50))
LANGUAGE plpgsql
AS $$
  BEGIN
  RETURN QUERY
  SELECT uczestnicy.iduczestnika, imie, nazwisko FROM uczestnicy
  INNER JOIN uczestnicydnia ON
uczestnicy.iduczestnika=uczestnicydnia.iduczestnika
  INNER JOIN uczestnicywarsztatow ON uczestnicydnia.iduczestnicydnia =
uczestnicywarsztatow.iduczestnicydnia
  INNER JOIN rezerwacjewarsztatow ON
uczestnicywarsztatow.idrezerwacjewarsztatow =
rezerwacjewarsztatow.idrezerwacjewarsztatow
  INNER JOIN warsztaty ON rezerwacjewarsztatow.idwarsztatu =
warsztaty.idwarsztatu
  WHERE warsztaty.idwarsztatu = uczestnicy_warsztatu.idwarsztatu
  ORDER BY uczestnicy.iduczestnika;
  END;
$$;
```

4.6 Funkcje pomocnicze do constraintów

```
CREATE OR REPLACE FUNCTION wolne_miejsca_dnia(id_dnia INTEGER)
RETURNS INTEGER
LANGUAGE plpgsql
AS $$
  DECLARE wszystkiemiejsca INTEGER;
  DECLARE zajetemiejsca INTEGER;
  DECLARE wolnemiejsca INTEGER;
  BEGIN
    wszystkiemiejsca := (SELECT iloscmiejsc FROM dzienkonferencji
                         WHERE iddniakonferencji = id_dnia);
    zajetemiejsca := (SELECT sum(iloscmiejsc) FROM rezerwacjedni
                      WHERE (iddnikonferencji = id_dnia AND dataanulowania
IS NULL));
    wolnemiejsca := (wszystkiemiejsca - zajetemiejsca);
    RETURN wolnemiejsca;
  END;
$$;
```

```
CREATE OR REPLACE FUNCTION wolne_miejsca_warsztatu(id_warsztatu INTEGER)
RETURNS INTEGER
LANGUAGE plpgsql
AS $$
  DECLARE wszystkiemiejsca INTEGER;
  DECLARE zajetemiejsca INTEGER;
  DECLARE wolnemiejsca INTEGER;
  BEGIN
    wszystkiemiejsca := (SELECT iloscmiejsc FROM warsztaty
                         WHERE idwarsztatu = id_warsztatu);
    zajetemiejsca := (SELECT sum(iloscmiejsc) FROM rezerwacjewarsztatow
                      WHERE (idwarsztatu = id_warsztatu AND
rezerwacjewarsztatow.dataanulowania IS NULL));
    wolnemiejsca := (wszystkiemiejsca - zajetemiejsca);
    RETURN wolnemiejsca;
  END;
$$;
```

```
CREATE OR REPLACE FUNCTION sprawdz_zgodnosc_rezerwacji(id_rezerwacjidnia
INTEGER)
RETURNS BOOLEAN
LANGUAGE plpgsql
AS $$
  DECLARE czy_oplacona INTEGER;
  DECLARE miejsca_zarezerwowane INTEGER;
  DECLARE uczestnicy_rezerwacji INTEGER;
  BEGIN
    czy_oplacona := (SELECT idoplaty FROM oplaty
                     WHERE idrezerwacjedni=id_rezerwacjidnia);
    IF czy_oplacona IS NULL THEN
      RAISE 'Rezerwacja jeszcze nie oplacona';
      RETURN FALSE;
    END IF;
    miejsca_zarezerwowane := (SELECT iloscmiejsc FROM rezerwacjedni
                              WHERE idrezerwacjedni = id_rezerwacjidnia);
    uczestnicy_rezerwacji := (SELECT COUNT(iduczestnicydnia) FROM
uczestnicydnia
                              HAVING idrezerwacjedni = id_rezerwacjidnia);
    IF miejsca_zarezerwowane != uczestnicy_rezerwacji THEN
      RAISE 'Nie wczyscy się jeszcze zapisali';
      RETURN FALSE;
    ELSE
      RETURN TRUE;
    END IF;
  END;
$$;
```

5. Triggery

Mała liczba triggerów, wynika z tego, ze większość warunków jest sprawdzana w constraintach ustawionych przy tworzeniu tabel.

5.1 Trigger sprawdź godziny warsztatów

Trigger uruchamia się przy wstawianu lub aktualizacji rekordu do tabeli Warsztaty i sprawdza czy godzina zakończenia jest po godzinie rozpoczęcia.

```
CREATE TRIGGER tr_sprawdz_godziny_warsztatow
AFTER INSERT OR UPDATE ON warsztaty
FOR EACH ROW EXECUTE PROCEDURE sprawdz_godziny_warsztatu();
```

5.2 Trigger anuluj rezerwacje warsztatu

Trigger uruchamia się po zmianie w tabeli RezerwacjeDni. Jeżeli rezerwacja dnia zostania anulowana to odpowiadająca jej rezerwacja warsztatu w tabelia RezerwacjeWarsztatów też musi być anulowana.

```
CREATE TRIGGER tr_anuluj_rezerwacje_warsztatu
AFTER UPDATE ON rezerwacjedni
FOR EACH ROW EXECUTE PROCEDURE anuluj_rezerwacje_warsztatu();
```

6. Indeksy

W celu optymalnego działania bazy danych utworzyliśmy indeksy na kolumnach, z których często korzystamy. W naszym systemie, indeksy są na kolumnach które są kluczem głównym oraz na kolumnach, w których jest constraint unique. Są one przedstawione graficznie na schemacie bazy.

7. Role w systemie

9.1 Administrator systemu – posiada dostęp do całej bazy danych, oraz ma możliwość modyfikacji .

9.2 Programista systemu – posiada dostęp do bazy danych poprzez widoki i procedury, nie ma możliwości modyfikacji.

9.3 Organizator konferencji – posiada dostęp do dodawania nowych konferencji i warsztatów jak również generowania wszelkich raportów.

9.4 Klient – ma dostęp do tworzenia nowych rezerwacji, uzupełniania danych o uczestnikach, oraz anulowania rezerwacji.

8. Generator danych

Do projektu został dołączony kod w języku python.