02 E3 Project

Information Systems and Databases

Instituto Superior Técnico

January 3, 2023

Group 2:

- 66325 Tomás Marques Videira Fonseca (100%) tomas.mvf@gmail.com
- 96135 Afonso Brito Caiado Correia Alemão (100%) afonso.alemao@tecnico.ulisboa.pt
- 96317 Rui Pedro Canário Daniel (100%) ruipcdaniel@tecnico.ulisboa.pt

Teachers:

- Flávio Martins
- Alessandro Gianola
- Francisco Regateiro

Lab Shift number: PB03

Project E3

Database Schema

1. The Relational Model

The following relational model is a database schema for the information system of a dental clinic, inspired by what you modeled in Part 1 of the project.

Relational Model

client(VAT, name, birth_date, street, city, zip, gender)

phone_number_client(VAT, phone)

> VAT: FK(client)

employee(VAT, name, birth_date, street, city, zip, IBAN, salary)

> IC: All employees are either receptionists, nurses or doctors
> IC: IBAN is a candidate key
> IC: Salary is a positive number

phone_number_employee(VAT, phone)

> VAT: FK(employee)

receptionist(VAT)

> VAT: FK(employee)

nurse(VAT)

> VAT: FK(employee)

doctor(VAT, specialization, biography, email)

> VAT: FK(employee)
> IC: All doctors are either trainees or permanent
> IC: Email is a candidate key

permanent_doctor(VAT, years)

> VAT: FK(doctor)

trainee_doctor(VAT, supervisor)

> VAT: FK(doctor)
> supervisor: FK(permanent_doctor)

supervision_report(VAT, date_timestamp, description, evaluation)

> VAT: FK(trainee_doctor)
> IC: evaluation is a number in the range from 1 to 5

appointment(VAT_doctor, date_timestamp, VAT_client, description)

> VAT_doctor: FK(doctor)
> VAT_client: FK(client)

consultation(VAT_doctor, date_timestamp, SOAP_S, SOAP_O, SOAP_A, SOAP_P)

> VAT_doctor, date_timestamp: FK(appointment)
> IC: Consultations are always assigned to at least one assistant nurse

consultation_assistant(VAT_doctor, date_timestamp, VAT_nurse)

> VAT_doctor, date_timestamp: FK(consultation)
> VAT_nurse: FK(nurse)

diagnostic_code(ID, description)

diagnostic_code_relation(ID1, ID2, type)

> ID1: FK(diagnostic_code)
> ID2: FK(diagnostic_code)

consultation_diagnostic(VAT_doctor, date_timestamp, ID)

> VAT_doctor, date_timestamp: FK(consultation)
> ID: FK(diagnostic_code)

medication(name, lab)

prescription(VAT_doctor, date_timestamp, ID, name, lab, dosage, description)

> VAT_doctor, date_timestamp, ID: FK(consultation_diagnostic)
> name, lab: FK(medication)

procedure(name, type)

procedure_in_consultation(name, VAT_doctor, date_timestamp, description)

> name: FK(procedure)
> VAT_doctor, date_timestamp: FK(consultation)

teeth(quadrant, number, name)

procedure_charting(name, VAT_doctor, date_timestamp, quadrant, number, desc, measure)

> name, VAT_doctor, date_timestamp: FK(procedure_in_consultation)
> quadrant, number: FK(teeth)

procedure_imaging(name, VAT_doctor, date_timestamp, file)

> name, VAT_doctor, date_timestamp: FK(procedure_in_consultation)

2. The Database Schema

For the relational model above, write the SQL instructions to create the database in the PostgreSQL database server. You should choose the most appropriate SQL data types for each column.

You can create the database `db` in Postgres using the instructions in Lab 01.

In [ ]:
```
%load_ext sql
%sql postgresql+psycopg://db:db@postgres/db
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

Connecting and switching to connection postgresql+psycopg://db:db@postgres/db

In [ ]:
```
%%sql

/* Drop all tables */
DROP VIEW IF EXISTS facts_consultations;
DROP VIEW IF EXISTS dim_date;
DROP VIEW IF EXISTS dim_client;
DROP VIEW IF EXISTS dim_location;
DROP TABLE IF EXISTS procedure_imaging;
DROP TABLE IF EXISTS procedure_charting;
DROP TABLE IF EXISTS teeth;
DROP TABLE IF EXISTS procedure_in_consultation;
DROP TABLE IF EXISTS procedure;
DROP TABLE IF EXISTS prescription;
DROP TABLE IF EXISTS medication;
```

```sql
DROP TABLE IF EXISTS consultation_diagnostic;
DROP TABLE IF EXISTS diagnostic_code_relation;
DROP TABLE IF EXISTS diagnostic_code;
DROP TABLE IF EXISTS consultation_assistant;
DROP TABLE IF EXISTS consultation;
DROP TABLE IF EXISTS appointment;
DROP TABLE IF EXISTS supervison_report;
DROP TABLE IF EXISTS trainee_doctor;
DROP TABLE IF EXISTS permanent_doctor;
DROP TABLE IF EXISTS doctor;
DROP TABLE IF EXISTS nurse;
DROP TABLE IF EXISTS receptionist;
DROP TABLE IF EXISTS phone_number_employee;
DROP TABLE IF EXISTS employee;
DROP TABLE IF EXISTS phone_number_client;
DROP TABLE IF EXISTS client;

CREATE TABLE client(
    VAT VARCHAR(20),
    name VARCHAR(80) NOT NULL,
    birth_date DATE NOT NULL,
    street VARCHAR(255) NOT NULL,
    city VARCHAR(30) NOT NULL,
    zip VARCHAR(12) NOT NULL,
    gender CHAR(1) NOT NULL,
    PRIMARY KEY(VAT),
    CHECK(LENGTH(zip) >= 2)
);

CREATE TABLE phone_number_client(
    VAT VARCHAR(20),
    phone VARCHAR(15),
    PRIMARY KEY(VAT, phone),
    FOREIGN KEY(VAT) REFERENCES client(VAT),
    CHECK(LENGTH(phone) >= 3)
);

CREATE TABLE employee(
    VAT VARCHAR(20),
    name VARCHAR(80) NOT NULL,
    birth_date DATE NOT NULL,
    street VARCHAR(255) NOT NULL,
    city VARCHAR(30) NOT NULL,
    zip VARCHAR(12) NOT NULL,
    IBAN VARCHAR(30) NOT NULL,
    salary NUMERIC(16,4) NOT NULL,
    PRIMARY KEY(VAT),
    UNIQUE(IBAN),
    CHECK(salary > 0),
    CHECK(LENGTH(zip) >= 2)
    /* -- No Employee can exist at the same time in both the table 'nurse'
    and in the table 'doctor' */
    /* -- No Employee can exist at the same time in both the table 'receptionist'
    and in the table 'doctor' */
    /* -- No Employee can exist at the same time in both the table 'nurse' and
    in the table 'receptionist' */
    /* -- Every Employee must exist either in the table 'nurse' or in the table 'doctor'
    or in the table 'receptionist' */
);

CREATE TABLE phone_number_employee(
    VAT VARCHAR(20),
    phone VARCHAR(15),
    PRIMARY KEY(VAT, phone),
    FOREIGN KEY(VAT) REFERENCES employee(VAT),
    CHECK(LENGTH(phone) >= 3)
);

CREATE TABLE receptionist(
    VAT VARCHAR(20),
    PRIMARY KEY(VAT),
    FOREIGN KEY(VAT) REFERENCES employee(VAT)
);

CREATE TABLE nurse(
    VAT VARCHAR(20),
    PRIMARY KEY(VAT),
    FOREIGN KEY(VAT) REFERENCES employee(VAT)
);

CREATE TABLE doctor(
    VAT VARCHAR(20),
    specialization VARCHAR(200) NOT NULL,
    biography TEXT NOT NULL,
    email VARCHAR(254) NOT NULL,
    PRIMARY KEY(VAT),
    FOREIGN KEY(VAT) REFERENCES employee(VAT),
    UNIQUE(email),
    CHECK(LENGTH(email) >= 6)
    /* -- No Doctor can exist at the same time in both the table 'permanent_doctor'
    and in the table 'trainee_doctor' */
    /* -- Every Doctor must exist either in the table 'permanent_doctor' or
    in the table 'trainee_doctor' */
);
```

```sql
CREATE TABLE permanent_doctor(
    VAT VARCHAR(20),
    years INTEGER NOT NULL,
    PRIMARY KEY(VAT),
    FOREIGN KEY(VAT) REFERENCES doctor(VAT),
    CHECK(years >= 0)
);

CREATE TABLE trainee_doctor(
    VAT VARCHAR(20),
    supervisor VARCHAR(20) NOT NULL,
    PRIMARY KEY(VAT),
    FOREIGN KEY(VAT) REFERENCES doctor(VAT),
    FOREIGN KEY(supervisor) REFERENCES permanent_doctor(VAT)

);

CREATE TABLE supervison_report(
    VAT VARCHAR(20),
    date_timestamp TIMESTAMP,
    description TEXT NOT NULL,
    evaluation NUMERIC(3,2) NOT NULL,
    PRIMARY KEY(VAT, date_timestamp),
    FOREIGN KEY(VAT) REFERENCES trainee_doctor(VAT),
    CHECK(evaluation >= 1 and evaluation <= 5)
);

CREATE TABLE appointment(
    VAT_doctor VARCHAR(20),
    date_timestamp TIMESTAMP,
    VAT_client VARCHAR(20) NOT NULL,
    description TEXT NOT NULL,
    PRIMARY KEY(VAT_doctor, date_timestamp),
    FOREIGN KEY(VAT_doctor) REFERENCES doctor(VAT),
    FOREIGN KEY(VAT_client) REFERENCES client(VAT)
);

CREATE TABLE consultation(
    VAT_doctor VARCHAR(20),
    date_timestamp TIMESTAMP,
    SOAP_S TEXT NOT NULL,
    SOAP_O TEXT NOT NULL,
    SOAP_A TEXT NOT NULL,
    SOAP_P TEXT NOT NULL,
    PRIMARY KEY(VAT_doctor, date_timestamp),
    FOREIGN KEY(VAT_doctor, date_timestamp)
        REFERENCES appointment(VAT_doctor, date_timestamp)
    /* -- Consultations are always assigned to at least one assistant nurse */
);

CREATE TABLE consultation_assistant(
    VAT_doctor VARCHAR(20),
    date_timestamp TIMESTAMP,
    VAT_nurse VARCHAR(20) NOT NULL,
    PRIMARY KEY(VAT_doctor, date_timestamp),
    FOREIGN KEY(VAT_doctor, date_timestamp)
        REFERENCES consultation(VAT_doctor, date_timestamp),
    FOREIGN KEY(VAT_nurse) REFERENCES nurse(VAT)
);

CREATE TABLE diagnostic_code(
    ID VARCHAR(7),
    description TEXT NOT NULL,
    PRIMARY KEY(ID)
);

CREATE TABLE diagnostic_code_relation(
    ID1 VARCHAR(7),
    ID2 VARCHAR(7),
    type VARCHAR(200) NOT NULL,
    PRIMARY KEY(ID1, ID2),
    FOREIGN KEY(ID1) REFERENCES diagnostic_code(ID),
    FOREIGN KEY(ID2) REFERENCES diagnostic_code(ID)
);

CREATE TABLE consultation_diagnostic(
    VAT_doctor VARCHAR(20),
    date_timestamp TIMESTAMP,
    ID VARCHAR(7),
    PRIMARY KEY(VAT_doctor, date_timestamp, ID),
    FOREIGN KEY(VAT_doctor, date_timestamp)
        REFERENCES consultation(VAT_doctor, date_timestamp),
    FOREIGN KEY(ID) REFERENCES diagnostic_code(ID)
);

CREATE TABLE medication(
    name VARCHAR(255),
    lab VARCHAR(200),
    PRIMARY KEY(name, lab)
);

CREATE TABLE prescription(
    VAT_doctor VARCHAR(20),
```

```
    date_timestamp TIMESTAMP,
    ID VARCHAR(7),
    name VARCHAR(255),
    lab VARCHAR(200),
    dosage VARCHAR(80) NOT NULL,
    description TEXT NOT NULL,
    PRIMARY KEY(VAT_doctor, date_timestamp, ID, name, lab),
    FOREIGN KEY(VAT_doctor, date_timestamp, ID)
        REFERENCES consultation_diagnostic(VAT_doctor, date_timestamp, ID),
    FOREIGN KEY(name, lab) REFERENCES medication(name, lab)

);

CREATE TABLE procedure(
    name VARCHAR(200),
    type VARCHAR(150) NOT NULL,
    PRIMARY KEY(name)
);

CREATE TABLE procedure_in_consultation(
    name VARCHAR(200),
    VAT_doctor VARCHAR(20),
    date_timestamp TIMESTAMP,
    description TEXT NOT NULL,
    PRIMARY KEY(name, VAT_doctor, date_timestamp),
    FOREIGN KEY(name) REFERENCES procedure(name),
    FOREIGN KEY(VAT_doctor, date_timestamp)
        REFERENCES consultation(VAT_doctor, date_timestamp)

);

CREATE TABLE teeth(
    quadrant CHAR(1),
    number CHAR(1),
    name VARCHAR(200) NOT NULL,
    PRIMARY KEY(quadrant, number)
);

CREATE TABLE procedure_charting(
    name VARCHAR(200),
    VAT_doctor VARCHAR(20),
    date_timestamp TIMESTAMP,
    quadrant CHAR(1),
    number CHAR(1),
    description TEXT NOT NULL,
    measure NUMERIC(6,2) NOT NULL,
    PRIMARY KEY(name, VAT_doctor, date_timestamp, quadrant, number),
    FOREIGN KEY(name, VAT_doctor, date_timestamp)
        REFERENCES procedure_in_consultation(name, VAT_doctor, date_timestamp),
    FOREIGN KEY(quadrant, number) REFERENCES teeth(quadrant, number)

);

CREATE TABLE procedure_imaging(
    name VARCHAR(200),
    VAT_doctor VARCHAR(20),
    date_timestamp TIMESTAMP,
    file TEXT,
    PRIMARY KEY(name, VAT_doctor, date_timestamp, file),
    FOREIGN KEY(name, VAT_doctor, date_timestamp)
        REFERENCES procedure_in_consultation(name, VAT_doctor, date_timestamp)

);
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

Out[ ]:

3. Populate the Database

Write a SQL script to populate the tables of the relational database with meaningful records of your choice, that you should design to ensure that we can validate the answers to the next questions.

In [ ]:

```
%%sql

INSERT INTO client (VAT, name, birth_date, street, city, zip, gender)
VALUES
    ('123456789', 'John Doe', '1990-05-15', '123 Pink St', 'New York', '1006119', 'M'),
    ('987654321', 'Margarida Corceiro', '2002-10-26', '456 Viana da Mota St', 'Lisbon', '9001001', 'F'),
    ('537450341', 'Marylin Monroe', '1985-10-20', ' St', 'Los Angeles', '9005561', 'F'),
    ('567890123', 'Tobey Maguire', '1995-03-07', '789 Oak St', 'Chicago', '6060551', 'M'),
    ('345678901', 'Tomás Fonseca', '1980-12-12', '101 Pine St', 'Lisbon', '7700552', 'M'),
    ('904444567', 'Rui Daniel', '2001-07-12', '210 Estalagem St', 'Rosário', '3315581', 'M'),
    ('905234567', 'Afonso Alemão', '2001-06-12', '210 Viana da Mota St', 'Lisbon', '3310551', 'M'),
    ('234567850', 'Júlio Paisana', '1969-04-20', '777 Elm St', 'Bora Bora', '9413301', 'M'),
    ('457054321', 'Billie Eilish', '1999-11-06', '456 Viana da Mota St', 'Lisbon', '9001005', 'F'),
    ('357652351', 'Mariana Almeida', '2001-06-13', '333 Big St', 'Mexico City', '9035521', 'F'),
    ('255152351', 'Sofia Almeida', '2002-10-26', '456 Viana da Mota St', 'Lisbon', '8201201', 'F'),
    ('181657351', 'Jessica Silva', '1985-10-20', '11 Carlos Mardel St', 'Coimbra', '2304521', 'F'),
    ('182797366', 'Mantorras', '1982-03-18', '11 Lisboa St', 'Lisbon', '2324522', 'M'),
    ('3083334733', 'Luis Vieira', '1960-03-04', '121 Estalagem St', 'Prisa', '10104', 'M');

INSERT INTO employee (VAT, name, birth_date, street, city, zip, IBAN, salary)
```

```
VALUES
    ('9121512141', 'Jane Sweettooth', '1992-08-20', '789 Oak St', 'Chicago', '60601', '123656889', 4000.00),
    ('7222324262', 'David Smith', '1985-03-15', '456 Elm St', 'Los Angeles', '90001', '987654321', 2000.00),
    ('0135012541', 'Emily Davis', '1980-11-10', '123 Main St', 'New York', '10001', '563890123', 4300.00),
    ('7322220202', 'Dolores Aveiro', '1977-06-12', '127 Main St', 'New York', '10003', '537093123', 5700.00),
    ('2032062803', 'Stone Cold', '1970-07-10', '128 Main St', 'Lisbon', '10004', '267830223', 2800.00),
    ('7423225472', 'Ed Sheeran', '1977-02-12', '127 Main St', 'New York', '10003', '537891123', 5700.00),
    ('2042462003', 'Tate McRae', '2001-03-17', '128 Estalagem St', 'Rosário', '10004', '367835223', 2800.00),
    ('3043042703', 'Cristiano Ronaldo', '2005-03-04', '128 Estalagem St', 'Rosário', '10004', '367330223', 280000.00),
    ('3083334733', 'Luis Vieira', '1960-03-04', '121 Estalagem St', 'Prisa', '10104', '317311123', 900.00);

INSERT INTO receptionist (VAT)
VALUES ('3043042703');

INSERT INTO nurse (VAT)
VALUES
    ('2042462003'),
    ('7423225472');

INSERT INTO doctor (VAT, specialization, biography, email)
VALUES
    ('9121512141', 'Orthodontics', 'Dr. Sweettooth specializes in Orthodontics.', 'sweet@gmail.com'),
    ('7222324262', 'Endodontics', 'Dr. Smith is passionate about children health
        and has been practicing pediatrics for over a decade.', 'smith@gmail.com'),
    ('0135012541', 'Periodontics', 'Dr. Davis specializes in Periodontics.', 'davis@gmail.com'),
    ('7322220202', 'Prosthodontics', 'Dr. Aveiro loves multi-tasking', 'aveiro@gmail.com'),
    ('2032062803', 'Pediatric Dentistry', 'Dr. Cold is dedicated to understanding Pediatric Dentistry.',
        'cold@gmail.com'),
    ('3083334733', 'Pediatric Dentistry', 'Dr. Vieira is dedicated to understanding Pediatric Dentistry.',
        'presi@gmail.com');


INSERT INTO permanent_doctor (VAT, years)
VALUES
    ('7322220202', 10),
    ('2032062803', 5),
    ('0135012541', 7),
    ('3083334733', 9);


INSERT INTO trainee_doctor (VAT, supervisor)
VALUES
    ('9121512141', '7322220202'),
    ('7222324262', '2032062803');

INSERT INTO supervison_report (VAT, date_timestamp, description, evaluation)
VALUES
    ('9121512141', '2023-01-15 09:00:00', 'Insufficient performance.', 3),
    ('9121512141', '2023-02-10 10:30:00', 'Bad performance', 1),
    ('7222324262', '2023-01-20 11:15:00', 'Sufficient diagnostic skills observed.', 3),
    ('7222324262', '2023-03-05 14:00:00', 'Excellent diagnostic skills observed.', 5);

INSERT INTO phone_number_client (VAT, phone)
VALUES
    ('123456789', '912345678'),
    ('987654321', '939876543'),
    ('537450341', '953745034'),
    ('567890123', '956789012'),
    ('345678901', '934567890'),
    ('345678901', '914464852'),
    ('345678901', '924367491'),
    ('904444567', '990444456'),
    ('905234567', '990523456'),
    ('234567850', '923456785'),
    ('457054321', '945705432'),
    ('357652351', '935765235'),
    ('255152351', '925515235'),
    ('255152351', '963455631'),
    ('181657351', '918165735');

INSERT INTO phone_number_employee (VAT, phone)
VALUES
    ('9121512141', '919121512'),
    ('9121512141', '919121513'),
    ('7222324262', '972223242'),
    ('0135012541', '901350125'),
    ('7322220202', '973222020'),
    ('2032062803', '920320628'),
    ('7423225472', '917423225'),
    ('7423225472', '917423323'),
    ('2042462003', '920424620'),
    ('3043042703', '913043042'),
    ('3083334733', '929292929');

INSERT INTO appointment (VAT_doctor, date_timestamp, VAT_client, description)
VALUES
    ('9121512141', '2023-12-15 10:00:00', '123456789', 'Regular checkup'),
    ('7222324262', '2023-12-16 11:00:00', '987654321', 'Regular checkup'),
    ('0135012541', '2023-12-17 09:00:00', '567890123', 'Regular checkup'),
    ('7322220202', '2023-12-18 13:00:00', '345678901', 'Regular checkup'),
    ('2032062803', '2023-12-19 14:00:00', '345678901', 'Surgery'),
    ('9121512141', '2023-11-15 10:00:00', '345678901', 'Regular checkup'),
    ('2032062803', '2019-11-07 10:00:00', '357652351', 'Regular checkup'),
    ('9121512141', '2023-11-19 11:00:00', '904444567', 'Regular checkup'),
```

```
    ('9121512141', '2023-11-21 09:00:00', '255152351', 'Surgery'),
    ('9121512141', '2023-11-23 08:00:00', '345678901', 'Regular checkup'),
    ('7222324262', '2023-11-26 11:00:00', '457054321', 'Regular checkup'),
    ('7222324262', '2019-11-26 11:00:00', '234567850', 'Regular checkup'),
    ('0135012541', '2019-11-17 09:00:00', '345678901', 'Surgery'),
    ('7322220202', '2023-11-28 13:00:00', '181657351', 'Regular checkup'),
    ('2032062803', '2023-11-29 14:00:00', '357652351', 'Regular checkup'),
    ('2032062803', '2019-11-29 14:00:00', '123456789', 'Regular checkup'),
    ('2032062803', '2019-01-01 14:00:00', '181657351', 'Daily checkup'),
    ('2032062803', '2019-01-02 14:00:00', '181657351', 'Daily checkup'),
    ('2032062803', '2019-01-03 14:00:00', '181657351', 'Daily checkup');

INSERT INTO consultation (VAT_doctor, date_timestamp, SOAP_S, SOAP_O, SOAP_A, SOAP_P)
VALUES
    ('9121512141', '2023-11-15 10:00:00', 'gingivitis', 'Objective2', 'Assessment23', 'Plan5'),
    ('2032062803', '2019-11-07 10:00:00', 'Subjective4', 'periodontitis', 'Assessment25', 'Plan6'),
    ('9121512141', '2023-11-19 11:00:00', 'Subjective2', 'Objective22', 'Assessment6', 'Plan1'),
    ('9121512141', '2023-11-21 09:00:00', 'Subjective1', 'Objective11', 'Assessment1', 'Plan2'),
    ('9121512141', '2023-11-23 08:00:00', 'periodontitis', 'Objective4', 'Assessment15', 'Plan3'),
    ('7222324262', '2019-11-26 11:00:00', 'Subjective2', 'Objective5', 'Assessment2', 'Plan4'),
    ('7222324262', '2023-11-26 11:00:00', 'Subjective2', 'Objective5', 'Assessment2', 'Plan4'),
    ('0135012541', '2023-12-17 09:00:00', 'Subjective1', 'Objective10', 'gingivitis', 'Plan5'),
    ('7322220202', '2023-11-28 13:00:00', 'Subjective4', 'Objective8', 'Assessment3', 'Plan2'),
    ('2032062803', '2023-11-29 14:00:00', 'Subjective5', 'Objective9', 'Assessment3', 'gingivitis'),
    ('0135012541', '2019-11-17 09:00:00', 'Subjective5', 'Objective9', 'Assessment3', 'gingivitis');

INSERT INTO diagnostic_code (ID, description)
VALUES
    ('110011D', 'Dental Cavities'),
    ('110021D', 'Gingivitis'),
    ('110031D', 'Caries'),
    ('110041D', 'Active dental caries'),
    ('110051D', 'Poor Oral Hygiene'),
    ('110061D', 'Overbite'),
    ('110071D', 'Infectious Disease');

INSERT INTO diagnostic_code_relation (ID1, ID2, type)
VALUES
    ('110011D', '110021D', 'Related-To'),
    ('110021D', '110011D', 'Related-To'),
    ('110041D', '110051D', 'Related-To'),
    ('110051D', '110041D', 'Related-To');

INSERT INTO consultation_diagnostic (VAT_doctor, date_timestamp, ID)
VALUES
    ('9121512141', '2023-11-15 10:00:00', '110011D'),
    ('9121512141', '2023-11-15 10:00:00', '110071D'),
    ('2032062803', '2019-11-07 10:00:00', '110021D'),
    ('9121512141', '2023-11-23 08:00:00', '110021D'),
    ('0135012541', '2019-11-17 09:00:00', '110011D'),
    ('0135012541', '2023-12-17 09:00:00', '110011D'),
    ('0135012541', '2023-12-17 09:00:00', '110071D'),
    ('0135012541', '2023-12-17 09:00:00', '110031D'),
    ('2032062803', '2023-11-29 14:00:00', '110011D'),
    ('7222324262', '2019-11-26 11:00:00', '110051D'),
    ('7222324262', '2023-11-26 11:00:00', '110041D'),
    ('7222324262', '2023-11-26 11:00:00', '110071D'),
    ('7222324262', '2023-11-26 11:00:00', '110021D');

INSERT INTO medication (name, lab)
VALUES
    ('Amoxicillin', 'GenericLab'),
    ('Ibuprofen', 'PainReliefCo'),
    ('Chlorhexidine', 'MouthCareInc'),
    ('Paracetamol', 'MouthCareInc'),
    ('Listerine', 'MouthCareInc'),
    ('Elugel', 'MouthCareInc');

INSERT INTO teeth (quadrant, number, name)
VALUES
    ('1', '2', 'Upper Right Lateral Incisor'),
    ('2', '1', 'Upper Left Central Incisor'),
    ('2', '2', 'Upper Left Lateral Incisor'),
    ('3', '1', 'Lower Left Central Incisor'),
    ('3', '2', 'Lower Left Lateral Incisor'),
    ('4', '1', 'Lower Right Central Incisor'),
    ('4', '2', 'Lower Right Lateral Incisor');

INSERT INTO procedure (name, type)
VALUES
    ('Tooth Extraction', 'Surgical'),
    ('Dental Implant', 'Surgical'),
    ('Teeth Cleaning', 'Non-Surgical'),
    ('Root Canal', 'Non-Surgical'),
    ('Cavity Filling', 'Non-Surgical'),
    ('Filling', 'Non-Surgical'),
    ('Cleaning', 'Non-Surgical'),
    ('Braces Adjustment', 'Non-Surgical'),
    ('CT Scan', 'Non-Surgical'),
    ('MRI', 'Non-Surgical'),
    ('Panoramic X-ray', 'Non-Surgical');

INSERT INTO procedure_in_consultation (name, VAT_doctor, date_timestamp, description)
VALUES
```

```
    ('Tooth Extraction', '9121512141', '2023-11-15 10:00:00', 'Extraction of wisdom tooth due to impaction'),
    ('Dental Implant', '7222324262', '2019-11-26 11:00:00', 'Implant placement in the upper right quadrant'),
    ('Filling', '0135012541', '2023-12-17 09:00:00', 'Routine dental cleaning to remove plaque and calculus'),
    ('Cleaning', '2032062803', '2019-11-07 10:00:00', 'Root canal treatment on lower left molar'),
    ('Braces Adjustment', '7222324262', '2023-11-26 11:00:00', 'Filling of caries on upper left premolar'),
    ('CT Scan', '2032062803', '2023-11-29 14:00:00', 'CT SCAN'),
    ('MRI', '2032062803', '2023-11-29 14:00:00', 'MRI'),
    ('Panoramic X-ray', '2032062803', '2023-11-29 14:00:00', 'Pan');

INSERT INTO procedure_charting (name, VAT_doctor, date_timestamp, quadrant, number, description, measure)
VALUES
    ('Filling', '0135012541', '2023-12-17 09:00:00', '2', '1', 'Filling for cavity', 5.0),
    ('Filling', '0135012541', '2023-12-17 09:00:00', '3', '1', 'Filling for cavity', 6.0),
    ('Cleaning', '2032062803', '2019-11-07 10:00:00', '1', '2', 'Teeth cleaning', 6.0),
    ('Braces Adjustment', '7222324262', '2023-11-26 11:00:00', '3', '1', 'Routine braces adjustment', 3.0),
    ('Braces Adjustment', '7222324262', '2023-11-26 11:00:00', '2', '1', 'Routine braces adjustment', 4.5);

INSERT INTO procedure_imaging (name, VAT_doctor, date_timestamp, file)
VALUES
    ('CT Scan', '2032062803', '2023-11-29 14:00:00', 'ctscan1.png'),
    ('MRI', '2032062803', '2023-11-29 14:00:00', 'mri1.png'),
    ('Panoramic X-ray', '2032062803', '2023-11-29 14:00:00', 'panoramic_xray.png');

INSERT INTO prescription (VAT_doctor, date_timestamp, ID, name, lab, dosage, description)
VALUES
    ('9121512141', '2023-11-15 10:00:00', '110011D', 'Amoxicillin', 'GenericLab', '500mg every 8 hours',
        'Antibiotic for gingivitis'),
    ('9121512141', '2023-11-15 10:00:00', '110071D', 'Ibuprofen', 'PainReliefCo', '200mg as needed',
        'Pain relief for dental pain'),
    ('9121512141', '2023-11-15 10:00:00', '110011D', 'Chlorhexidine', 'MouthCareInc', 'Use twice daily',
        'Mouthwash for periodontitis'),
    ('2032062803', '2019-11-07 10:00:00', '110021D', 'Chlorhexidine', 'MouthCareInc', 'Use twice daily',
        'Mouthwash for periodontitis'),
    ('0135012541', '2019-11-17 09:00:00', '110011D', 'Amoxicillin', 'GenericLab', '500mg every 8 hours',
        'Antibiotic for gingivitis'),
    ('7222324262', '2019-11-26 11:00:00', '110051D', 'Ibuprofen', 'PainReliefCo', '200mg as needed',
        'Pain relief for dental pain'),
    ('7222324262', '2023-11-26 11:00:00', '110041D', 'Ibuprofen', 'PainReliefCo', '200mg as needed',
        'Pain relief for dental pain'),
    ('7222324262', '2023-11-26 11:00:00', '110071D', 'Chlorhexidine', 'MouthCareInc', 'Use twice daily',
        'Mouthwash for periodontitis'),
    ('0135012541', '2019-11-17 09:00:00', '110011D', 'Paracetamol', 'MouthCareInc', '200mg as needed',
        'Pain relief for dental pain'),
    ('7222324262', '2023-11-26 11:00:00', '110021D', 'Listerine', 'MouthCareInc', '200mg as needed',
        'Pain relief for dental pain'),
    ('0135012541', '2019-11-17 09:00:00', '110011D', 'Elugel', 'MouthCareInc', '200mg as needed',
        'Pain relief for dental pain');
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

14 rows affected.

9 rows affected.

1 rows affected.

2 rows affected.

6 rows affected.

4 rows affected.

2 rows affected.

4 rows affected.

15 rows affected.

11 rows affected.

19 rows affected.

11 rows affected.

7 rows affected.

4 rows affected.

13 rows affected.

6 rows affected.

7 rows affected.

11 rows affected.

8 rows affected.

5 rows affected.

3 rows affected.

11 rows affected.

Out[ ]:

Web Application

Using Views for a Dashboard

Create views over the tables in the database model, corresponding to the following relational schema.

dim_date(date, day, month, year)

IC: date corresponds to a date existing in consultations

dim_client(VAT, gender, age)

VAT: FK(client)

dim_location(zip, city)

IC: zip corresponds to a zip code existing in clients

facts_consultations(VAT, date, zip, num_diagnostic_codes, num_procedures)

VAT: FK(dim_client)

date: FK(dim_date)

zip: FK(dim_location)

Present the SQL code for creating each of the views corresponding to the tables in the previous model, so that the views feature the corresponding records in the database (i.e., information on all the clients that had consultations, together with the associated number of procedures, number of diagnostic codes).

In [ ]:
```
%%sql
DROP VIEW IF EXISTS dim_date;

CREATE VIEW dim_date AS
SELECT DISTINCT
    date_timestamp AS date,
    EXTRACT(DAY FROM date_timestamp) AS day,
    EXTRACT(MONTH FROM date_timestamp) AS month,
    EXTRACT(YEAR FROM date_timestamp) AS year
FROM consultation;
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

Out[ ]:

In [ ]:
```
%%sql
SELECT *
FROM dim_date;
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

11 rows affected.

Out[ ]:

date	day	month	year
2023-12-17 09:00:00	17	12	2023
2023-11-26 11:00:00	26	11	2023
2023-11-28 13:00:00	28	11	2023
2019-11-26 11:00:00	26	11	2019
2019-11-07 10:00:00	7	11	2019
2023-11-15 10:00:00	15	11	2023
2019-11-17 09:00:00	17	11	2019
2023-11-23 08:00:00	23	11	2023
2023-11-21 09:00:00	21	11	2023
2023-11-29 14:00:00	29	11	2023
2023-11-19 11:00:00	19	11	2023

In [ ]:
```
%%sql
DROP VIEW IF EXISTS dim_client;

CREATE VIEW dim_client AS
SELECT
    VAT,
    gender,
    EXTRACT(YEAR FROM age(birth_date)) AS age
FROM client;
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

Out[ ]:

In [ ]:
```
%%sql
SELECT *
FROM dim_client;
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

14 rows affected.

Out[ ]:

vat	gender	age
123456789	M	33
987654321	F	21
537450341	F	38
567890123	M	28
345678901	M	43
904444567	M	22
905234567	M	22
234567850	M	54
457054321	F	24
357652351	F	22
255152351	F	21
181657351	F	38
182797366	M	41
3083334733	M	63

In [ ]:
```
%%sql
DROP VIEW IF EXISTS dim_location;

CREATE VIEW dim_location AS
SELECT DISTINCT
    zip,
    city
FROM client;
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

Out[ ]:

In [ ]:
```
%%sql

SELECT *
FROM dim_location;
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

14 rows affected.

Out[ ]:

zip	city
6060551	Chicago
2324522	Lisbon
9005561	Los Angeles
2304521	Coimbra
10104	Prisa
9001005	Lisbon
7700552	Lisbon
8201201	Lisbon
9413301	Bora Bora
9001001	Lisbon
3315581	Rosário
1006119	New York
3310551	Lisbon
9035521	Mexico City

In [ ]:
```
%%sql

DROP VIEW IF EXISTS facts_consultations;

CREATE VIEW facts_consultations AS
SELECT
    c.VAT,
    d.date,
    l.zip,
    (SELECT COUNT(cd.ID)
        FROM consultation_diagnostic cd
        WHERE cd.date_timestamp = cc.date_timestamp
            AND cd.VAT_doctor = cc.VAT_doctor) AS num_diagnostic_codes,
    (SELECT COUNT(pc.name)
        FROM procedure_in_consultation pc
        WHERE pc.date_timestamp = cc.date_timestamp
            AND pc.VAT_doctor = cc.VAT_doctor) AS num_procedures
FROM dim_client c
    JOIN client cl ON cl.VAT = c.VAT
    JOIN dim_location l ON cl.zip = l.zip
```

```
    JOIN appointment a ON a.VAT_client = c.VAT
    JOIN consultation cc ON cc.date_timestamp = a.date_timestamp
        AND cc.VAT_doctor = a.VAT_doctor
    JOIN dim_date d ON d.date = cc.date_timestamp;
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

Out[ ]:

In [ ]:
```
%%sql
SELECT *
FROM facts_consultations
ORDER BY (num_procedures, num_diagnostic_codes) DESC;
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

11 rows affected.

Out[ ]:

vat	date	zip	num_diagnostic_codes	num_procedures
357652351	2023-11-29 14:00:00	9035521	1	3
567890123	2023-12-17 09:00:00	6060551	3	1
457054321	2023-11-26 11:00:00	9001005	3	1
345678901	2023-11-15 10:00:00	7700552	2	1
357652351	2019-11-07 10:00:00	9035521	1	1
234567850	2019-11-26 11:00:00	9413301	1	1
345678901	2019-11-17 09:00:00	7700552	1	0
345678901	2023-11-23 08:00:00	7700552	1	0
255152351	2023-11-21 09:00:00	8201201	0	0
904444567	2023-11-19 11:00:00	3315581	0	0
181657351	2023-11-28 13:00:00	2304521	0	0

Indexes

Suggest indexes that could improve the performance of the following query:

In [ ]:
```
SELECT
    VAT,
    COUNT(*) AS num_consultations,
    SUM(num_procedures) AS total_procedures
FROM
    facts_consultations
GROUP BY
    VAT
ORDER BY
    total_procedures;
```

Provide SQL instructions for implementing the most appropriate indexes. Justify your choice and provide the corresponding query plan(s).

In [ ]:
```
%config SqlMagic.displaylimit = None
```

displaylimit: Value None will be treated as 0 (no limit)

Considering the attributes involved in the "facts_consultations" query, we created indexes to significantly enhance data access speed because they enable the location of an item without the need to search everywhere.

The considered attributes are as follows:

- client.VAT, client.zip
- consultation.VAT_doctor, consultation.date_timestamp
- appointment.VAT_client, appointment.date_timestamp, appointment.VAT_doctor
- consultation_diagnostic.VAT_doctor, consultation_diagnostic.date_timestamp, consultation_diagnostic.ID
- procedure_in_consultation.VAT_doctor, procedure_in_consultation.date_timestamp, procedure_in_consultation.name

For each entity, we created both composite indexes and individual indexes for each attribute. For example, an index for client.VAT, an index for client.zip, and a composite index for (client.VAT, client.zip).

We opted for MIN-MAX Indexes (BRIN), which are indexes that keep track of the places where not to search for a value, indicating the blocks that are not of interest and thus do not need further processing.

We also tested indexes using BTREE and HASH. However, BRIN exhibited the best performance, meaning a higher speedup compared to the baseline where there are no indexes.

In [ ]:
```
%%sql

DROP INDEX IF EXISTS idx_client;
DROP INDEX IF EXISTS idx_consult;
DROP INDEX IF EXISTS idx_app;
DROP INDEX IF EXISTS idx_cd;
DROP INDEX IF EXISTS idx_pc;
```

```
DROP INDEX IF EXISTS idx_client_vat;
DROP INDEX IF EXISTS idx_client_zip;
DROP INDEX IF EXISTS idx_consult_vat_doctor;
DROP INDEX IF EXISTS idx_consult_date_timestamp;
DROP INDEX IF EXISTS idx_app_vat_client;
DROP INDEX IF EXISTS idx_app_date_timestamp;
DROP INDEX IF EXISTS idx_app_vat_doctor;
DROP INDEX IF EXISTS idx_cd_vat_doctor;
DROP INDEX IF EXISTS idx_cd_date_timestamp;
DROP INDEX IF EXISTS idx_cd_id;
DROP INDEX IF EXISTS idx_pc_vat_doctor;
DROP INDEX IF EXISTS idx_pc_date_timestamp;
DROP INDEX IF EXISTS idx_pc_name;

-- Without indexes

EXPLAIN ANALYZE SELECT
    VAT,
    COUNT(*) AS num_consultations,
    SUM(num_procedures) AS total_procedures
FROM
    facts_consultations
GROUP BY
    VAT
ORDER BY
    total_procedures;
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

49 rows affected.

QUERY PLAN
Sort (cost=4555.54..4555.74 rows=80 width=98) (actual time=0.914..0.981 rows=8 loops=1)
Sort Key: (sum((SubPlan 1)))
Sort Method: quicksort Memory: 25kB
-> GroupAggregate (cost=133.91..4553.01 rows=80 width=98) (actual time=0.716..0.911 rows=8 loops=1)
Group Key: client.vat
-> Sort (cost=133.91..134.81 rows=360 width=124) (actual time=0.684..0.747 rows=11 loops=1)
Sort Key: client.vat
Sort Method: quicksort Memory: 25kB
-> Hash Join (cost=85.62..118.62 rows=360 width=124) (actual time=0.541..0.700 rows=11 loops=1)
Hash Cond: (cc.date_timestamp = d.date)
-> Hash Join (cost=57.72..77.22 rows=360 width=132) (actual time=0.399..0.519 rows=11 loops=1)
Hash Cond: ((a.vat_client)::text = (cl.vat)::text)
-> Hash Join (cost=19.00..35.71 rows=360 width=132) (actual time=0.083..0.149 rows=11 loops=1)
Hash Cond: ((a.date_timestamp = cc.date_timestamp) AND ((a.vat_doctor)::text = (cc.vat_doctor)::text))
-> Seq Scan on appointment a (cost=0.00..14.40 rows=440 width=124) (actual time=0.005..0.031 rows=19 loops=1)
-> Hash (cost=13.60..13.60 rows=360 width=66) (actual time=0.052..0.056 rows=11 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Seq Scan on consultation cc (cost=0.00..13.60 rows=360 width=66) (actual time=0.003..0.018 rows=11 loops=1)
-> Hash (cost=37.72..37.72 rows=80 width=116) (actual time=0.303..0.326 rows=14 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Hash Join (cost=25.60..37.72 rows=80 width=116) (actual time=0.193..0.299 rows=14 loops=1)
Hash Cond: ((cl.zip)::text = (l.zip)::text)
-> Hash Join (cost=11.80..22.82 rows=80 width=158) (actual time=0.056..0.116 rows=14 loops=1)
Hash Cond: ((client.vat)::text = (cl.vat)::text)
-> Seq Scan on client (cost=0.00..10.80 rows=80 width=58) (actual time=0.005..0.024 rows=14 loops=1)
-> Hash (cost=10.80..10.80 rows=80 width=100) (actual time=0.038..0.042 rows=14 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Seq Scan on client cl (cost=0.00..10.80 rows=80 width=100) (actual time=0.002..0.019 rows=14 loops=1)
-> Hash (cost=12.80..12.80 rows=80 width=42) (actual time=0.124..0.134 rows=14 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Subquery Scan on l (cost=11.20..12.80 rows=80 width=42) (actual time=0.048..0.107 rows=14 loops=1)
-> HashAggregate (cost=11.20..12.00 rows=80 width=120) (actual time=0.045..0.069 rows=14 loops=1)
Group Key: client_1.zip, client_1.city
Batches: 1 Memory Usage: 24kB
-> Seq Scan on client client_1 (cost=0.00..10.80 rows=80 width=120) (actual time=0.002..0.020 rows=14 loops=1)
-> Hash (cost=25.40..25.40 rows=200 width=8) (actual time=0.125..0.133 rows=11 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Subquery Scan on d (cost=19.90..25.40 rows=200 width=8) (actual time=0.065..0.111 rows=11 loops=1)
-> HashAggregate (cost=19.90..23.40 rows=200 width=104) (actual time=0.062..0.078 rows=11 loops=1)
Group Key: consultation.date_timestamp, EXTRACT(day FROM consultation.date_timestamp), EXTRACT(month FROM consultation.date_timestamp), EXTRACT(year FROM consultation.date_timestamp)
Batches: 1 Memory Usage: 40kB
-> Seq Scan on consultation (cost=0.00..16.30 rows=360 width=104) (actual time=0.018..0.035 rows=11 loops=1)
SubPlan 1
-> Aggregate (cost=12.25..12.26 rows=1 width=8) (actual time=0.007..0.009 rows=1 loops=11)
-> Seq Scan on procedure_in_consultation pc (cost=0.00..12.25 rows=1 width=418) (actual time=0.002..0.003 rows=1 loops=11)
Filter: ((date_timestamp = cc.date_timestamp) AND ((vat_doctor)::text = (cc.vat_doctor)::text))
Rows Removed by Filter: 7
Planning Time: 1.152 ms
Execution Time: 1.164 ms

In [ ]:
```
%%sql

DROP INDEX IF EXISTS idx_client;
DROP INDEX IF EXISTS idx_consult;
DROP INDEX IF EXISTS idx_app;
DROP INDEX IF EXISTS idx_cd;
```

```
DROP INDEX IF EXISTS idx_pc;

DROP INDEX IF EXISTS idx_client_vat;
DROP INDEX IF EXISTS idx_client_zip;
DROP INDEX IF EXISTS idx_consult_vat_doctor;
DROP INDEX IF EXISTS idx_consult_date_timestamp;
DROP INDEX IF EXISTS idx_app_vat_client;
DROP INDEX IF EXISTS idx_app_date_timestamp;
DROP INDEX IF EXISTS idx_app_vat_doctor;
DROP INDEX IF EXISTS idx_cd_vat_doctor;
DROP INDEX IF EXISTS idx_cd_date_timestamp;
DROP INDEX IF EXISTS idx_cd_id;
DROP INDEX IF EXISTS idx_pc_vat_doctor;
DROP INDEX IF EXISTS idx_pc_date_timestamp;
DROP INDEX IF EXISTS idx_pc_name;

CREATE INDEX idx_client ON client USING BRIN(VAT, zip);
CREATE INDEX idx_consult ON consultation USING BRIN(VAT_doctor, date_timestamp);
CREATE INDEX idx_app ON appointment USING BRIN(VAT_client, date_timestamp, VAT_doctor);
CREATE INDEX idx_cd ON consultation_diagnostic USING BRIN(VAT_doctor, date_timestamp, ID);
CREATE INDEX idx_pc ON procedure_in_consultation USING BRIN(VAT_doctor, date_timestamp, name);

CREATE INDEX idx_client_vat ON client USING BRIN(VAT);
CREATE INDEX idx_client_zip ON client USING BRIN(zip);
CREATE INDEX idx_consult_vat_doctor ON consultation USING BRIN(VAT_doctor);
CREATE INDEX idx_consult_date_timestamp ON consultation USING BRIN(date_timestamp);
CREATE INDEX idx_app_vat_client ON appointment USING BRIN(VAT_client);
CREATE INDEX idx_app_date_timestamp ON appointment USING BRIN(date_timestamp);
CREATE INDEX idx_app_vat_doctor ON appointment USING BRIN(VAT_doctor);
CREATE INDEX idx_cd_vat_doctor ON consultation_diagnostic USING BRIN(VAT_doctor);
CREATE INDEX idx_cd_date_timestamp ON consultation_diagnostic USING BRIN(date_timestamp);
CREATE INDEX idx_cd_id ON consultation_diagnostic USING BRIN(ID);
CREATE INDEX idx_pc_vat_doctor ON procedure_in_consultation USING BRIN(VAT_doctor);
CREATE INDEX idx_pc_date_timestamp ON procedure_in_consultation USING BRIN(date_timestamp);
CREATE INDEX idx_pc_name ON procedure_in_consultation USING BRIN(name);


EXPLAIN ANALYZE SELECT
    VAT,
    COUNT(*) AS num_consultations,
    SUM(num_procedures) AS total_procedures
FROM
    facts_consultations
GROUP BY
    VAT
ORDER BY
    total_procedures;
```

Running query in 'postgresql+psycopg://db:***@postgres/db'

47 rows affected.

QUERY PLAN
Sort (cost=21.84..21.87 rows=11 width=98) (actual time=0.783..0.853 rows=8 loops=1)
Sort Key: (sum((SubPlan 1)))
Sort Method: quicksort Memory: 25kB
-> HashAggregate (cost=21.51..21.65 rows=11 width=98) (actual time=0.750..0.819 rows=8 loops=1)
Group Key: client.vat
Batches: 1 Memory Usage: 24kB
-> Hash Join (cost=7.31..8.97 rows=11 width=124) (actual time=0.420..0.652 rows=11 loops=1)
Hash Cond: ((a.date_timestamp = cc.date_timestamp) AND ((a.vat_doctor)::text = (cc.vat_doctor)::text))
-> Hash Join (cost=6.04..7.64 rows=11 width=132) (actual time=0.366..0.560 rows=11 loops=1)
Hash Cond: ((cl.zip)::text = (l.zip)::text)
-> Hash Join (cost=4.37..5.82 rows=11 width=174) (actual time=0.234..0.385 rows=11 loops=1)
Hash Cond: ((client.vat)::text = (cl.vat)::text)
-> Hash Join (cost=3.06..4.47 rows=11 width=190) (actual time=0.182..0.294 rows=11 loops=1)
Hash Cond: ((a.vat_client)::text = (client.vat)::text)
-> Hash Join (cost=1.74..3.11 rows=11 width=132) (actual time=0.135..0.208 rows=11 loops=1)
Hash Cond: (a.date_timestamp = d.date)
-> Seq Scan on appointment a (cost=0.00..1.19 rows=19 width=124) (actual time=0.003..0.030 rows=19 loops=1)
-> Hash (cost=1.61..1.61 rows=11 width=8) (actual time=0.115..0.124 rows=11 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Subquery Scan on d (cost=1.30..1.61 rows=11 width=8) (actual time=0.045..0.092 rows=11 loops=1)
-> HashAggregate (cost=1.30..1.50 rows=11 width=104) (actual time=0.043..0.061 rows=11 loops=1)
Group Key: consultation.date_timestamp, EXTRACT(day FROM consultation.date_timestamp), EXTRACT(month FROM consultation.date_timestamp), EXTRACT(year FROM consultation.date_timestamp)
Batches: 1 Memory Usage: 24kB
-> Seq Scan on consultation (cost=0.00..1.19 rows=11 width=104) (actual time=0.007..0.025 rows=11 loops=1)
-> Hash (cost=1.14..1.14 rows=14 width=58) (actual time=0.042..0.045 rows=14 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Seq Scan on client (cost=0.00..1.14 rows=14 width=58) (actual time=0.002..0.022 rows=14 loops=1)
-> Hash (cost=1.14..1.14 rows=14 width=100) (actual time=0.041..0.045 rows=14 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Seq Scan on client cl (cost=0.00..1.14 rows=14 width=100) (actual time=0.002..0.022 rows=14 loops=1)
-> Hash (cost=1.49..1.49 rows=14 width=42) (actual time=0.125..0.133 rows=14 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Subquery Scan on l (cost=1.21..1.49 rows=14 width=42) (actual time=0.049..0.108 rows=14 loops=1)
-> HashAggregate (cost=1.21..1.35 rows=14 width=120) (actual time=0.045..0.068 rows=14 loops=1)
Group Key: client_1.zip, client_1.city
Batches: 1 Memory Usage: 24kB
-> Seq Scan on client client_1 (cost=0.00..1.14 rows=14 width=120) (actual time=0.003..0.022 rows=14 loops=1)
-> Hash (cost=1.11..1.11 rows=11 width=66) (actual time=0.043..0.046 rows=11 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Seq Scan on consultation cc (cost=0.00..1.11 rows=11 width=66) (actual time=0.007..0.023 rows=11 loops=1)
SubPlan 1
-> Aggregate (cost=1.12..1.13 rows=1 width=8) (actual time=0.007..0.009 rows=1 loops=11)
-> Seq Scan on procedure_in_consultation pc (cost=0.00..1.12 rows=1 width=418) (actual time=0.002..0.003 rows=1 loops=11)
Filter: ((date_timestamp = cc.date_timestamp) AND ((vat_doctor)::text = (cc.vat_doctor)::text))
Rows Removed by Filter: 7
Planning Time: 1.076 ms
Execution Time: 0.971 ms

A Web Application Using the Database

Dashboard

dashboard/dashboard.html

In [ ]:
```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Dashboard{% endblock %}</h1>
{% endblock %}

{% block content %}
    <a href="/clients" class="button-link">
        <button type="button">Clients</button>
    </a>
    <a href="/new_client" class="button-link">
        <button type="button">New Client</button>
    </a>
    <table>
        <thead>
            <tr>
                <th>VAT Client</th>
                <th>Date</th>
                <th>Total Diagnostic Codes</th>
                <th>Total Procedures</th>
            </tr>
        </thead>
        <tbody>
            {% for fact in facts_consultations %}
                <tr>
                    <td>{{ fact.vat or "NULL" }}</td>
                    <td>{{ fact.date or "NULL"}}</td>
                    <td>{{ fact.total_diagnostic_codes }}</td>
                    <td>{{ fact.total_procedures }}</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% endblock %}
```

base.html

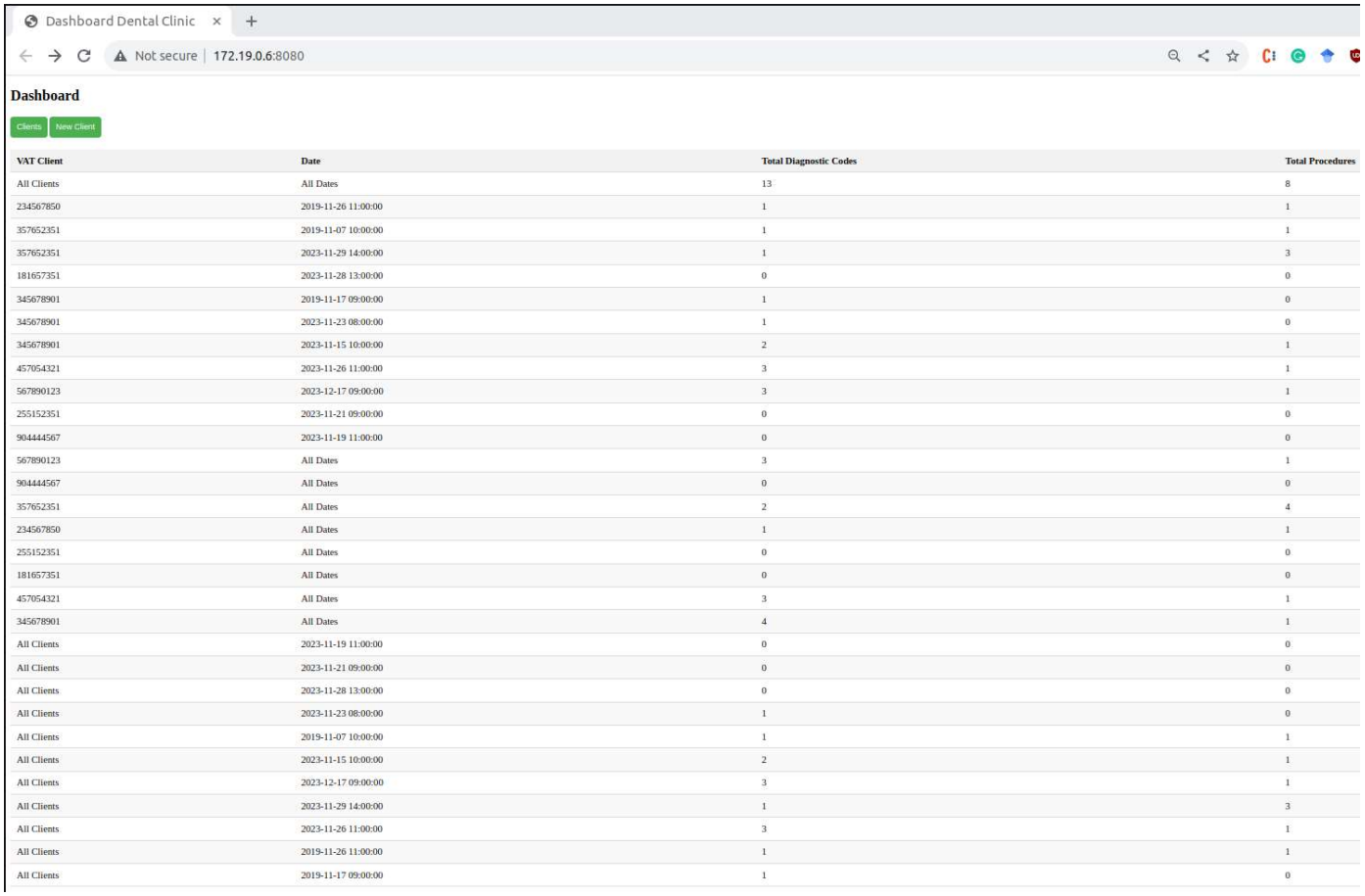
In [ ]:
```
<!doctype html>
<html>
<head>
    <title>{% block title %}{% endblock %} Dental Clinic </title>
</head>
<body>
    <section class="content">
        <header>
            {% block header %}{% endblock %}
        </header>
        {% for message in get_flashed_messages() %}
            <div class="flash">{{ message }}</div>
        {% endfor %}
        <div>
            {% block content %}{% endblock %}
        </div>
    </section>
</body>
<style>
    /* Style for the table */
    table {
        width: 100%;
        border-collapse: collapse;
        margin-top: 20px;
    }

    /* Style for table header */
    thead {
        background-color: #f2f2f2;
    }

    th, td {
        padding: 10px;
        text-align: left;
        border-bottom: 1px solid #ddd;
    }

    /* Style for alternating rows */
    tbody tr:nth-child(even) {
        background-color: #f9f9f9;
    }

    /* Style for buttons */
    .button-link {
        text-decoration: none;
    }

    button {
        padding: 10px;
        background-color: #4CAF50;
        color: white;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }
```

```
    button:hover {
        background-color: #45a049;
    }
</style>
</html>
```

Associated endpoints

In [ ]:
```
@app.route("/", methods=["GET"])
@app.route("/dashboard", methods=["GET"])
def dashboard():
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT VAT, Date, SUM (num_procedures) AS total_procedures,
                    SUM (num_diagnostic_codes) AS total_diagnostic_codes
                FROM facts_consultations
                GROUP BY CUBE (VAT, Date);
            """)
            facts_consultations = cur.fetchall()
            app.logger.debug(f"Found {len(facts_consultations)} rows.")

    return render_template("dashboard/dashboard.html", facts_consultations=facts_consultations)
```

Screenshot of Web page



VAT Client	Date	Total Diagnostic Codes	Total Procedures
All Clients	All Dates	13	8
234567850	2019-11-26 11:00:00	1	1
357652351	2019-11-07 10:00:00	1	1
357652351	2023-11-29 14:00:00	1	3
181657351	2023-11-28 13:00:00	0	0
345678901	2019-11-17 09:00:00	1	0
345678901	2023-11-23 08:00:00	1	0
345678901	2023-11-15 10:00:00	2	1
457054321	2023-11-26 11:00:00	3	1
567890123	2023-12-17 09:00:00	3	1
255152351	2023-11-21 09:00:00	0	0
904444567	2023-11-19 11:00:00	0	0
567890123	All Dates	3	1
904444567	All Dates	0	0
357652351	All Dates	2	4
234567850	All Dates	1	1
255152351	All Dates	0	0
181657351	All Dates	0	0
457054321	All Dates	3	1
345678901	All Dates	4	1
All Clients	2023-11-19 11:00:00	0	0
All Clients	2023-11-21 09:00:00	0	0
All Clients	2023-11-28 13:00:00	0	0
All Clients	2023-11-23 08:00:00	1	0
All Clients	2019-11-07 10:00:00	1	1
All Clients	2023-11-15 10:00:00	2	1
All Clients	2023-12-17 09:00:00	3	1
All Clients	2023-11-29 14:00:00	1	3
All Clients	2023-11-26 11:00:00	3	1
All Clients	2019-11-26 11:00:00	1	1
All Clients	2019-11-17 09:00:00	1	0

Description

Dashboard Web page that uses the facts_consultations View and OLAP queries. CUBE was used to obtain total diagnostic codes and total procedures grouped by {(), (VAT Client), (Date), (VAT Client, Date)} with one single query. The button "Clients" redirects to the clients space. The button "New Client" redirects to the page "Add Client" that allows the creation of a new client.

Add Client

clients/new_client.html

In [ ]:
```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Add Client {% endblock %}</h1>
{% endblock %}

{% block content %}
    <h2>Client</h2>

    <form id="addClientForm" method="post" action="/new_client2">
        <!-- VAT -->
```

```
        <label for="vat">VAT:</label>
        <input type="text" id="vat" name="vat"  maxlength="20" required>

        <!-- Name -->
        <label for="name">Name:</label>
        <input type="text" id="name" name="name"  maxlength="80" required>

        <!-- Birth Date -->
        <label for="birth_date">Birth Date:</label>
        <input type="date" id="birth_date" name="birth_date" required pattern="\d{4}-\d{2}-\d{2}">

        <!-- Street -->
        <label for="street">Street:</label>
        <input type="text" id="street" name="street"  maxlength="255" required>

        <!-- City -->
        <label for="city">City:</label>
        <input type="text" id="city" name="city"  maxlength="30" required>

        <!-- ZIP -->
        <label for="zip">ZIP:</label>
        <input type="text" id="zip" name="zip"  maxlength="12" required>

        <!-- Gender -->
        <label for="gender">Gender:</label>
        <select id="gender" name="gender" required>
            <option value="M">Male</option>
            <option value="F">Female</option>
            <option value="O">Other</option>
        </select>

        <!-- Submit button -->
        <button type="submit">Submit</button>
    </form>

    <!-- Button to redirect -->
    <button onclick="window.locatio
n.href='/dashbo
ad'">Back to Dashboard</button>
{% endblock %}
```

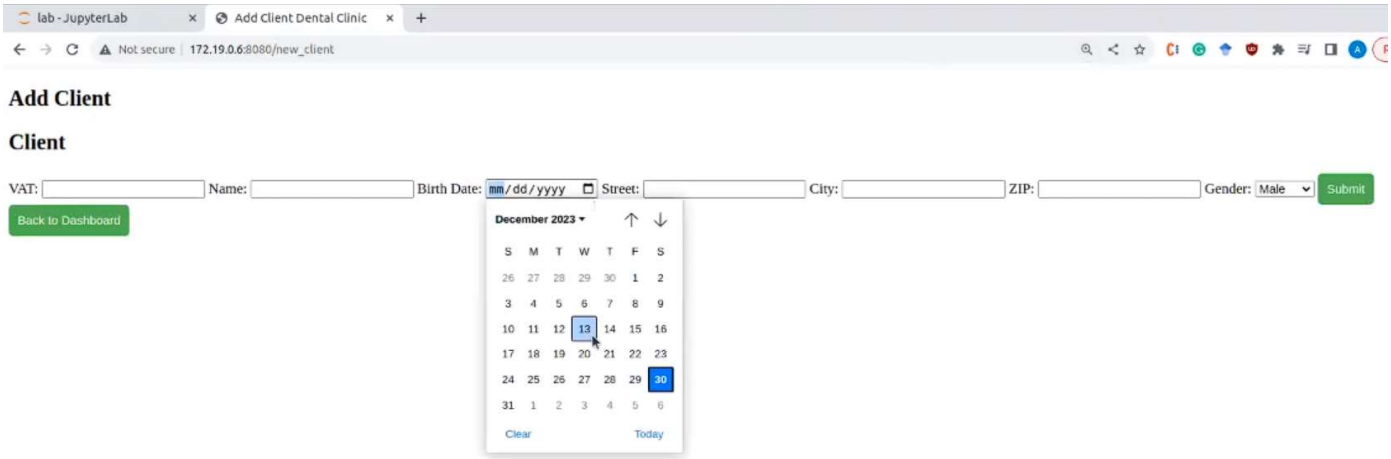
Associated endpoints

In [ ]:

```
def validate_date(date):
    try:
        # Try to parse date using the correct format
        datetime.strptime(date, '%Y-%m-%d')
        return True  # date is valid
    except ValueError:
        return False  # date is invalid

@app.route("/new_client", methods=["GET"])
def add_client_dashboard():

    return render_template("clients/new_client.html")

@app.route("/new_client2", methods=["POST"])
def add_client2():
    VAT = request.form.get("vat")
    name = request.form.get("name")
    birth_date = request.form.get("birth_date")
    street = request.form.get("street")
    city = request.form.get("city")
    gender = request.form.get("gender")
    zip_code = request.form.get("zip")

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT VAT
                FROM client;
                """)

            db_VAT_client = cur.fetchall()
            db_VAT_client = [row.vat for row in db_VAT_client]
            app.logger.debug(f"Found {cur.rowcount} db_VAT_client(s).")

    error = ""

    if VAT in db_VAT_client:
        error = 'VAT client already exists'

    if not validate_date(birth_date):
        error = "birthdate is invalid"

    if error != "":
        flash(error)
        return render_template("clients/new_client.html")

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                INSERT INTO client(VAT, name, birth_date, street, city, zip, gender)
```

```
                VALUES
                (%(VAT)s, %(name)s, %(birth_date)s, %(street)s, %(city)s, %(zip)s, %(gender)s);

            """, {"VAT": VAT, "name": name, "birth_date": birth_date, "street": street,
                  "city": city, "zip": zip_code, "gender": gender})

            conn.commit()

    flash('Client created successfully.')
    return redirect('/dashboard')
```

Screenshot of Web page



Description

Form to create a new client.

It is protected against:

- VAT already in the database;
- VAT length higher than 20 characters;
- name length higher than 80 characters;
- street length higher than 255 characters;
- city length higher than 30 characters;
- zip length higher than 12 characters;
- Invalid birth date.

All fields are required. "Back to Dashboard" redirects to "Dashboard" page. "Submit" button adds the client to the database and redirects to "Dashboard". If an error occurs in this process, the page refreshes adding an error message to its display.

Clients

clients/clients.html

In [ ]:

```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Clients{% endblock %}</h1>
{% endblock %}

{% block content %}
    <button onclick="window.location.href='/dashboard'">Back</button>
    <a href="/new_client" class="button-link">
        <button type="button">New Client</button>
    </a>

    <form action="{{ url_for('clients2') }}" method="post">
        <input type="text" name="search" placeholder="Search by VAT, Name, Address..." value="{{ request.form.get('search', '') }}">
        <button type="submit">Search</button>
    </form>

    <table>
        <thead>
            <tr>
                <th>VAT</th>
                <th>Name</th>
                <th>Birth Date</th>
                <th>Street</th>
                <th>City</th>
                <th>ZIP</th>
                <th>Gender</th>
                <th>Appoitments</th>
                <th>Create Appointment</th>
            </tr>
        </thead>
        <tbody>
```

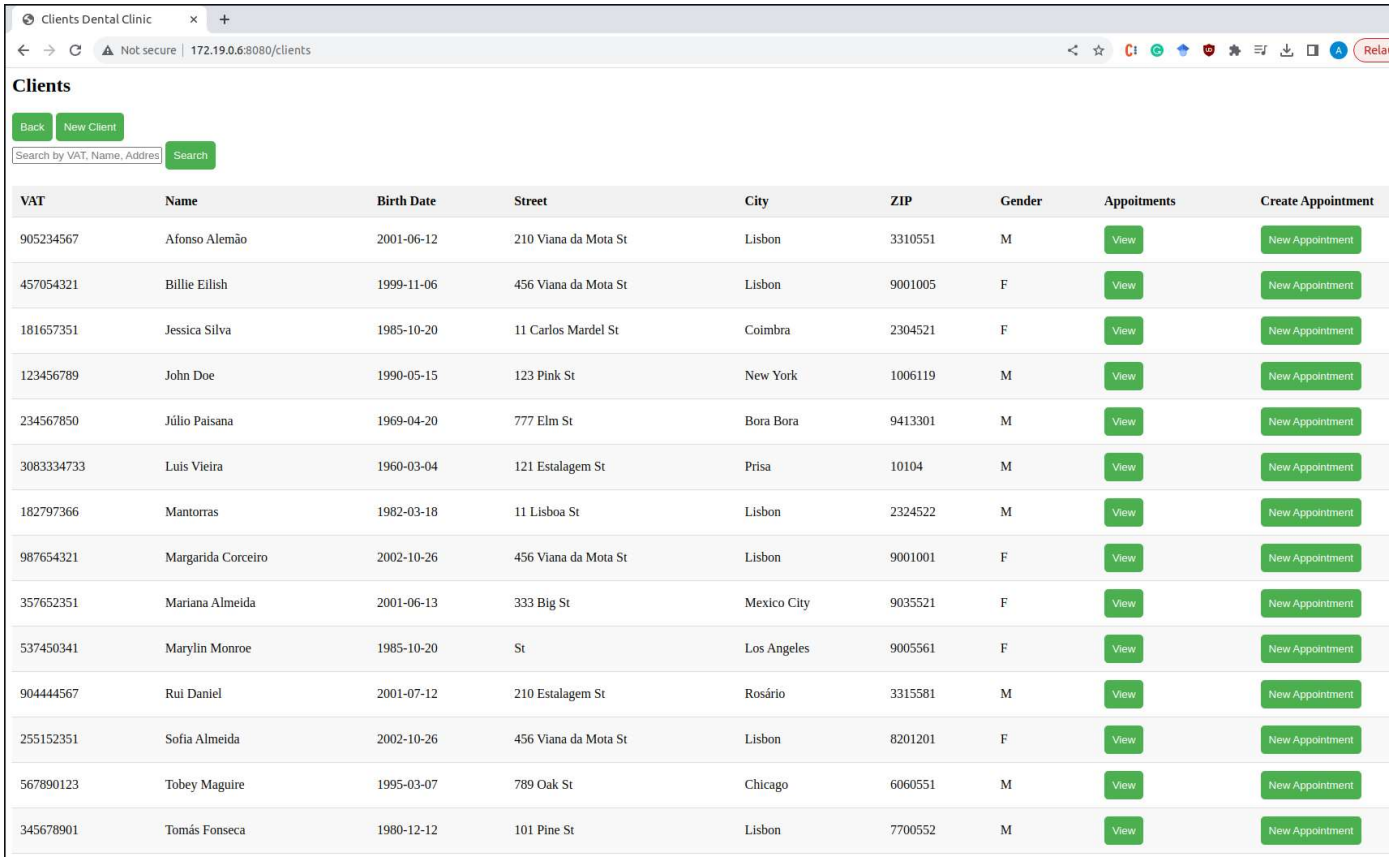
```
            {% for client in clients %}
                <tr>
                    <td>{{ client.vat }}</td>
                    <td>{{ client.name }}</td>
                    <td>{{ client.birth_date }}</td>
                    <td>{{ client.street }}</td>
                    <td>{{ client.city }}</td>
                    <td>{{ client.zip }}</td>
                    <td>{{ client.gender }}</td>
                    <td><a href="/client/{{ client.vat }}"><button type="button">View</button></a></td>
                    <td><a href="/client/{{ client.vat }}/new_appointment" class="button-link">
                            <button type="button">New Appointment</button>
                        </a>
                    </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% endblock %}
```

Associated endpoints

In [ ]:

```
@app.route("/clients", methods=["GET"])
def clients():
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
                FROM client
                ORDER BY name ASC;
            """)
            clients = cur.fetchall()
            app.logger.debug(f"Found {len(clients)} rows.")

    return render_template("clients/clients.html", clients=clients)

@app.route("/clients2", methods=["POST"])
def clients2():
    search = request.form.get("search")

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
                FROM client
                WHERE name ILIKE %(search_like)s
                OR VAT = %(search)s
                OR street ILIKE %(search_like)s
                OR city ILIKE %(search_like)s
                OR zip ILIKE %(search_like)s
                ORDER BY name ASC;
            """, {'search': search, 'search_like': '%' + search + '%'})
            clients = cur.fetchall()
            app.logger.debug(f"Found {len(clients)} rows.")

    return render_template("clients/clients.html", clients=clients)
```

Screenshot of Web page

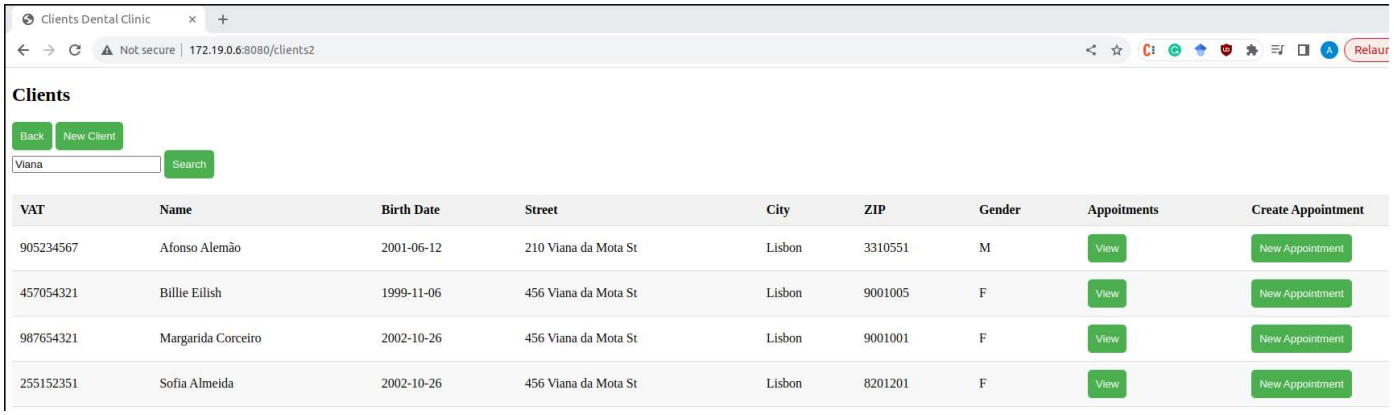


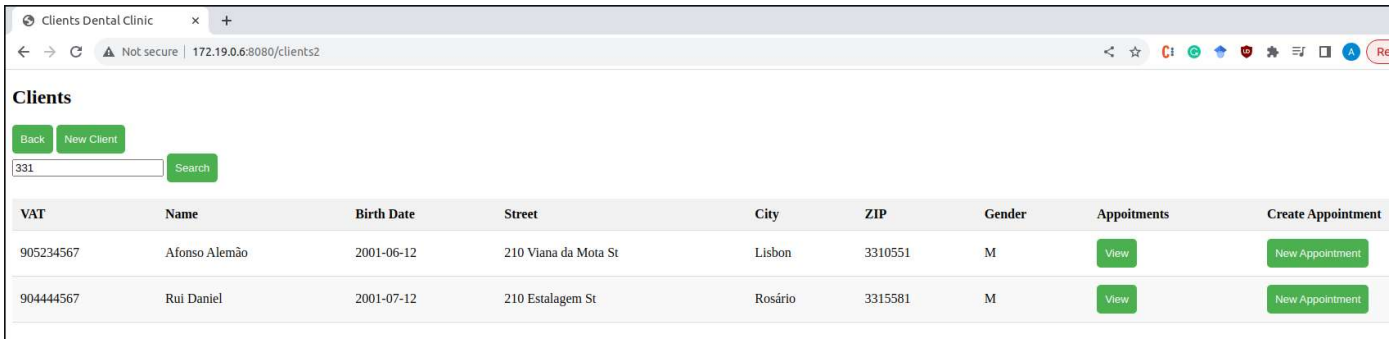
VAT	Name	Birth Date	Street	City	ZIP	Gender	Appoitments	Create Appointment
905234567	Afonso Alemão	2001-06-12	210 Viana da Mota St	Lisbon	3310551	M	View	New Appointment
457054321	Billie Eilish	1999-11-06	456 Viana da Mota St	Lisbon	9001005	F	View	New Appointment
181657351	Jessica Silva	1985-10-20	11 Carlos Mardel St	Coimbra	2304521	F	View	New Appointment
123456789	John Doe	1990-05-15	123 Pink St	New York	1006119	M	View	New Appointment
234567850	Júlio Paisana	1969-04-20	777 Elm St	Bora Bora	9413301	M	View	New Appointment
3083334733	Luis Vieira	1960-03-04	121 Estalagem St	Prisa	10104	M	View	New Appointment
182797366	Mantorras	1982-03-18	11 Lisboa St	Lisbon	2324522	M	View	New Appointment
987654321	Margarida Corceiro	2002-10-26	456 Viana da Mota St	Lisbon	9001001	F	View	New Appointment
357652351	Mariana Almeida	2001-06-13	333 Big St	Mexico City	9035521	F	View	New Appointment
537450341	Marylin Monroe	1985-10-20	St	Los Angeles	9005561	F	View	New Appointment
904444567	Rui Daniel	2001-07-12	210 Estalagem St	Rosário	3315581	M	View	New Appointment
255152351	Sofia Almeida	2002-10-26	456 Viana da Mota St	Lisbon	8201201	F	View	New Appointment
567890123	Tobey Maguire	1995-03-07	789 Oak St	Chicago	6060551	M	View	New Appointment
345678901	Tomás Fonseca	1980-12-12	101 Pine St	Lisbon	7700552	M	View	New Appointment

VAT	Name	Birth Date	Street	City	ZIP	Gender	Appoitments	Create Appointment
905234567	Afonso Alemão	2001-06-12	210 Viana da Mota St	Lisbon	3310551	M	View	New Appointment

VAT	Name	Birth Date	Street	City	ZIP	Gender	Appoitments	Create Appointment
905234567	Afonso Alemão	2001-06-12	210 Viana da Mota St	Lisbon	3310551	M	View	New Appointment

VAT	Name	Birth Date	Street	City	ZIP	Gender	Appoitments	Create Appointment
905234567	Afonso Alemão	2001-06-12	210 Viana da Mota St	Lisbon	3310551	M	View	New Appointment
457054321	Billie Eilish	1999-11-06	456 Viana da Mota St	Lisbon	9001005	F	View	New Appointment
182797366	Mantorras	1982-03-18	11 Lisboa St	Lisbon	2324522	M	View	New Appointment
987654321	Margarida Corceiro	2002-10-26	456 Viana da Mota St	Lisbon	9001001	F	View	New Appointment
255152351	Sofia Almeida	2002-10-26	456 Viana da Mota St	Lisbon	8201201	F	View	New Appointment
345678901	Tomás Fonseca	1980-12-12	101 Pine St	Lisbon	7700552	M	View	New Appointment





Web page to display existing clients in the database. The search bar allows matching clients based on different information elements: given the VAT, a (part of the) name for the client, and/or the (parts of the) address, you should display the records of matching clients.

For each displayed client, we included two options. One of them is the possibility of registering a new appointment: the button "New Appointment" redirects to "Add Appointment" page for that client. The other option is to view the client's corresponding appointments and consultations: the button "View" redirects to "Appointments and Consultations" page.

This page also includes a button "New Client" for adding new clients to the database, which redirects to "Add Client".

Add Appointment

clients/add_appointment.html

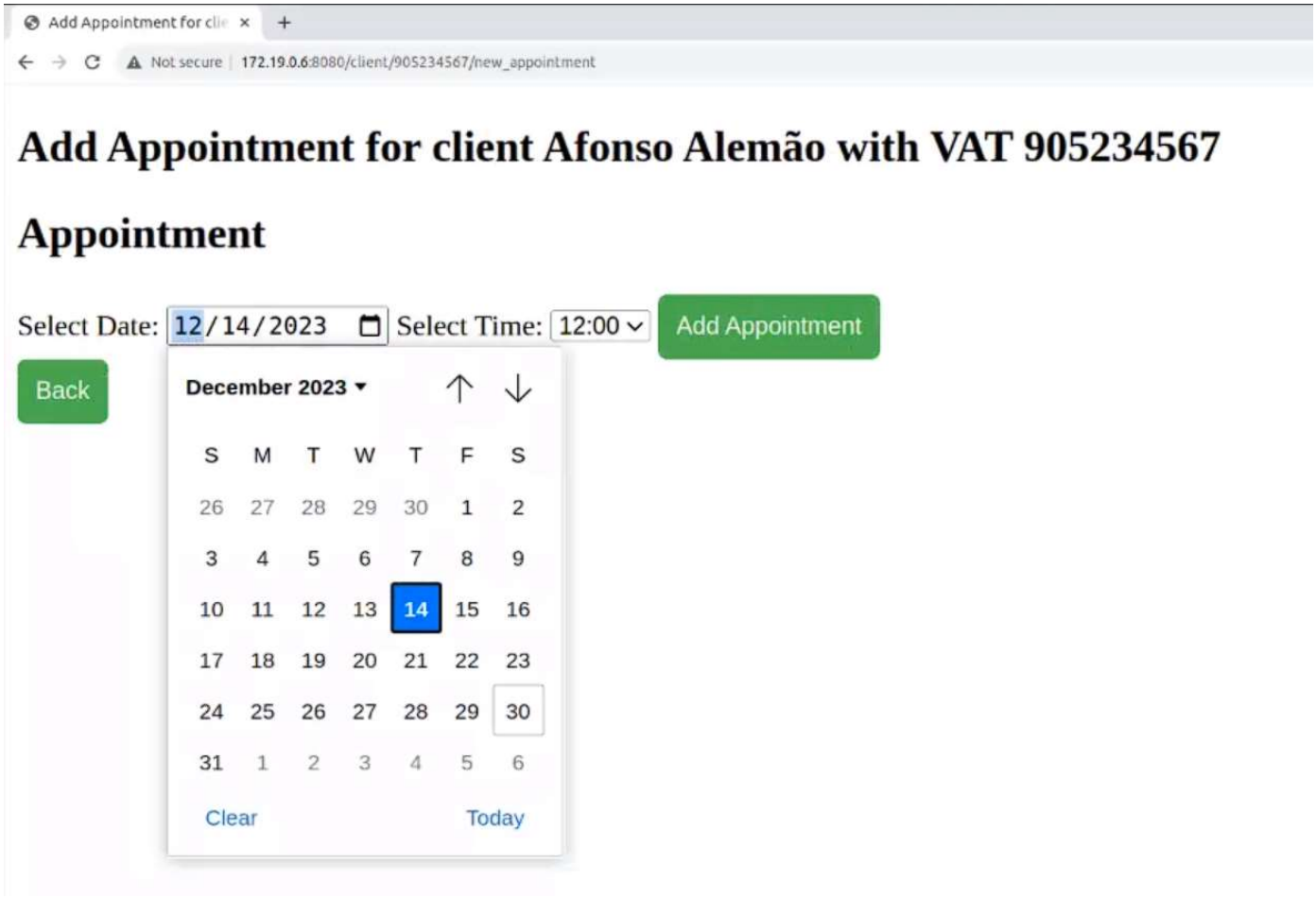
In [ ]:
```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Add Appointment for client {{ client.name }} with VAT {{ client.vat }}{% endblock %}</h1>
{% endblock %}

{% block content %}
    <h2>Appointment</h2>

    <form id="addAppointmentForm" method="post" action="/client/{{ client.vat }}/new_appointment_doctor">

        <label for="date">Select Date:</label>
        <input type="date" name="date" id="date" required pattern="\d{4}-\d{2}-\d{2}">

        <label for="time">Select Time:</label>
        <select name="time" id="time" required>
            {% for slot in available_slots %}
                <option value="{{ slot }}">{{ slot }}</option>
            {% endfor %}
        </select>

        <button type="submit">Add Appointment</button>
    </form>

    <!-- Button to redirect -->
    <button onclick="window.location.href='/clients'">Back</button>
{% endblock %}
```

Associated endpoints

In [ ]:
```
@app.route("/client/<VAT>/new_appointment", methods=["GET"])
def add_appointment_dashboard(VAT):
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
                FROM client
                WHERE VAT = %(VAT)s;
            """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")
```

```
            cur.execute("""
                SELECT *
                FROM appointment;
            """)
            appointments = cur.fetchall()
            app.logger.debug(f"Found {cur.rowcount} appointment(s).")

    available_slots = ['9:00', '10:00', '11:00', '12:00', '13:00', '14:00', '15:00', '16:00', '17:00']
    return render_template("clients/add_appointment.html", client = client, appointments = appointments, available_slots = available_slots)
```

Screenshot of Web page



Description

Form to create a new appointment.

It is protected against invalid dates and times.

All fields are required. "Back" redirects to "Clients" page. "Add Appointment" button redirects to "Add Appointment Doctor" where the user will select between the available doctors. If an error occurs in this process, the page refreshes adding an error message to its display.

Add Appointment Doctor

clients/add_appointment_doctor.html

In [ ]:

```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Add Appointment for client {{ client.name }} with VAT {{ client.vat }}{% endblock %}</h1>
{% endblock %}

{% block content %}
    <h2>Doctor</h2>
    {% if doctors != [] %}
    <form id="addDoctorForm" method="post" action="/client/{{ client.vat }}/{{date_timestamp}}/new_appointment2">
        <label for="doctor">Choose a Doctor by VAT:</label>
        <select id="VAT_doctor" name="VAT_doctor">
            {% for doctor in doctors %}
                <option value="{{ doctor.vat }}">{{ doctor.vat }}</option>
            {% endfor %}
        </select>

        <label for="description">Description:</label>
        <input type="text" id="description" name="description" maxlength="65535" required>

        <button type="submit">Add Appointment</button>
```

```
    </form>

    <h2>Available Doctors:</h2>
    <table>
        <thead>
            <tr>
                <th>VAT</th>
                <th>Name</th>
                <th>Biography</th>
                <th>E-mail</th>
                <th>Specialization</th>
            </tr>
        </thead>
        <tbody>
            {% for doctor in doctors %}
                <tr>
                    <td>{{ doctor.vat }}</td>
                    <td>{{ doctor.name }}</td>
                    <td>{{ doctor.biography }}</td>
                    <td>{{ doctor.email }}</td>
                    <td>{{ doctor.specialization }}</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>

    {% else %}
        No doctors available!
    {% endif %}

    <!-- Button to redirect -->
    <button onclick="window.location.href='/clients'">Back to Clients</button>
{% endblock %}
```

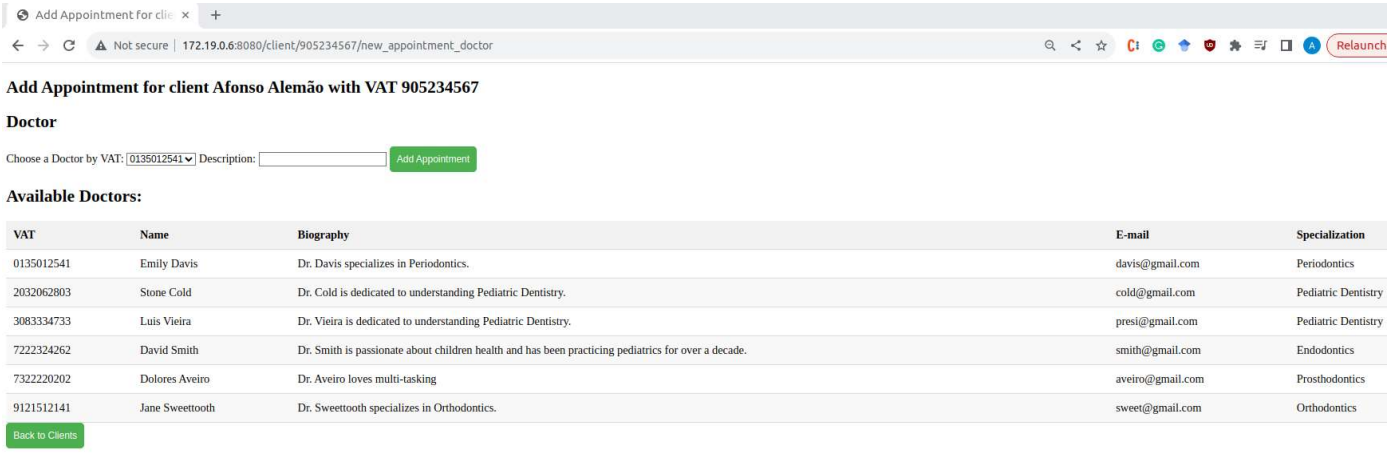
Associated endpoints

In [ ]:

```
@app.route("/client/<VAT>/new_appointment_doctor", methods=["POST"])
def add_appointment_doctor_dashboard(VAT):
    date = request.form.get("date")
    time = request.form.get("time") + ":00"
    datetime_str = date + " " + time
    format_str = "%Y-%m-%d %H:%M:%S"
    datetime_obj = datetime.strptime(datetime_str, format_str)
    date_timestamp = datetime.timestamp(datetime_obj)
    date_timestamp = datetime_obj.strftime(format_str)


    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
                FROM client
                WHERE VAT = %(VAT)s;
            """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")


            cur.execute("""
                SELECT e.name, d.specialization, d.email, d.biography, e.VAT
                FROM doctor AS d JOIN employee as e ON e.VAT = d.VAT
                WHERE d.VAT NOT IN(
                    SELECT d1.VAT
                    FROM doctor AS d1
                    JOIN appointment AS a ON a.VAT_doctor = d1.VAT
                    WHERE a.date_timestamp = %(date_timestamp)s
                )
                ORDER BY e.VAT;
            """, {"date_timestamp": date_timestamp})
            doctors = cur.fetchall()
            app.logger.debug(f"Found {cur.rowcount} doctor(s).")

    return render_template("clients/add_appointment_doctor.html", client = client, doctors = doctors, date_timestamp = date_timestamp)

@app.route("/client/<VAT>/<date_timestamp>/new_appointment2", methods=["POST"])
def add_appointment2(VAT, date_timestamp):
    VAT_doctor = request.form.get("VAT_doctor")
    description = request.form.get("description")
    error = ""

    if error != "":
        flash(error)

        return redirect('/client/' + VAT)

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""

                INSERT INTO appointment (VAT_doctor, date_timestamp, VAT_client, description)
                VALUES
                (%(VAT_doctor)s, %(date_timestamp)s, %(VAT_client)s, %(description)s);
```

```
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "description": description, "VAT_client": VAT})

            conn.commit()

    flash('Appointment created successfully.')
    return redirect('/client/' + VAT)
```

Screenshot of Web page



Description

Form to create a new appointment where a doctor will be selected between the available doctors at the previously selected date and time.

It is protected against:

- Invalid doctor's VAT;
- description length higher than 65535 characters and empty description.

All fields are required. "Back to Clients" redirects to "Clients" page. "Add Appointment" button adds the appointment to the database and redirects to "Appointments and Consultations". If an error occurs in this process, the page refreshes adding an error message to its display.

Appointments and Consultations

clients/client_vat.html

In [ ]:
```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Appointments and Consultations of Client {{ client.name }} with VAT {{ client.vat }}{% endblock %}</h1>
{% endblock %}

{% block content %}
    <button onclick="window.location.href='/clients'">Back</button>

    <h2>Appointments</h2>
    <table>
        <thead>
            <tr>
                <th>VAT Doctor</th>
                <th>Date and Time</th>
                <th>VAT Client</th>
                <th>Description</th>
                <th>View Consultation</th>
                <th>Create Consultation</th>
            </tr>
        </thead>
        <tbody>
            {% for appointment in appointments %}
                <tr>
                    <td>{{ appointment.vat_doctor }}</td>
                    <td>{{ appointment.date_timestamp }}</td>
                    <td>{{ appointment.vat_client }}</td>
                    <td>{{ appointment.description }}</td>
                    <td>
                        <a href="/client/{{ client.vat }}/{{ appointment.vat_doctor }}/{{ appointment.date_timestamp }}">
                            <button type="button">View</button>
                        </a>
                    </td>
                    {% if appointment.is_in_consultation == False %}
                    <td>
                        <a href="/client/{{ client.vat }}/{{ appointment.vat_doctor }}/{{ appointment.date_timestamp }}/create_consultation">
                            <button type="button">Create</button>
                        </a>
                    </td>
                    {% else %}
                    <td>
                        Already Created
```

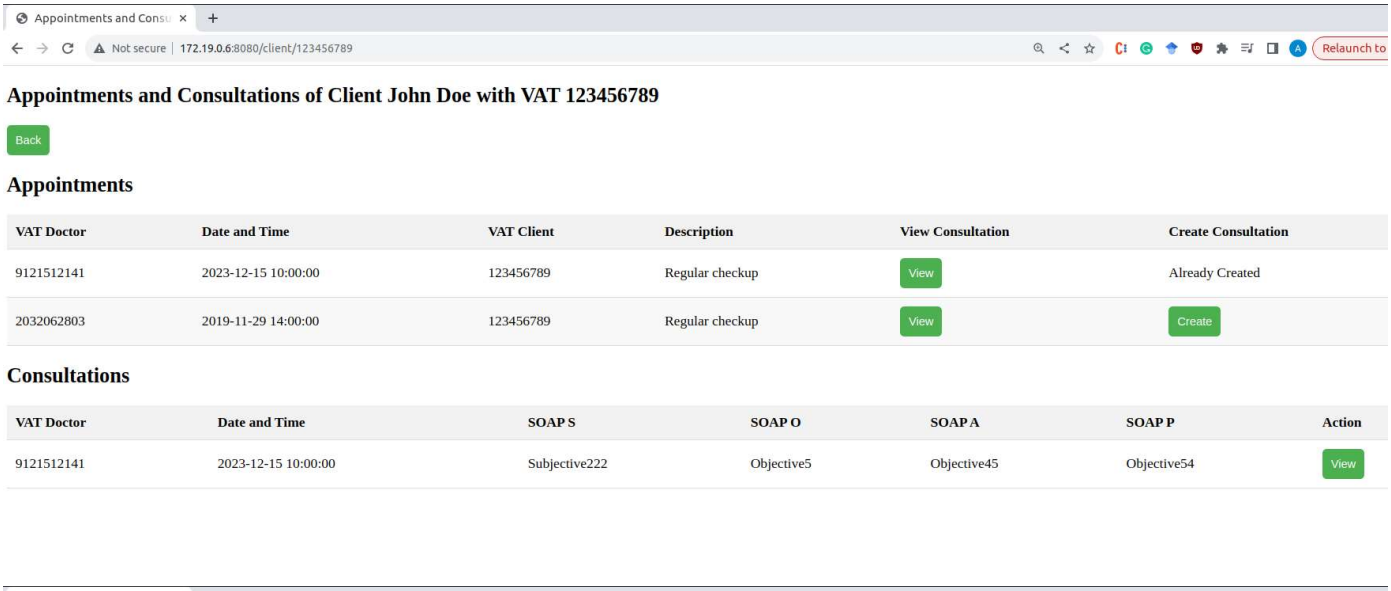
```
                    </td>
                    {% endif %}

                </tr>
            {% endfor %}
        </tbody>
    </table>

    <h2>Consultations</h2>
    <table>
        <thead>
            <tr>
                <th>VAT Doctor</th>
                <th>Date and Time</th>
                <th>SOAP S</th>
                <th>SOAP O</th>
                <th>SOAP A</th>
                <th>SOAP P</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            {% for consultation in consultations %}
                <tr>
                    <td>{{ consultation.vat_doctor }}</td>
                    <td>{{ consultation.date_timestamp }}</td>
                    <td>{{ consultation.soap_s }}</td>
                    <td>{{ consultation.soap_o }}</td>
                    <td>{{ consultation.soap_a }}</td>
                    <td>{{ consultation.soap_p }}</td>
                    <td>
                        <a href="/client/{{ client.vat }}/{{ consultation.vat_doctor }}/{{ consultation.date_timestamp }}">
                            <button type="button">View</button>
                        </a>
                    </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% endblock %}
```

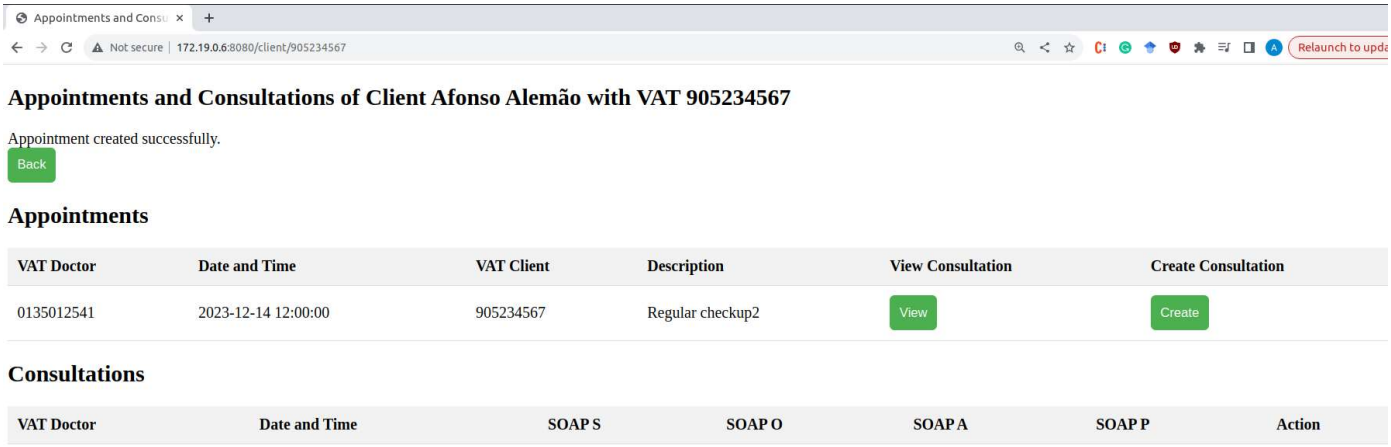
Associated endpoints

In [ ]:

```
@app.route("/client/<VAT>", methods=["GET"])
def client_vat(VAT):

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
                FROM client
                WHERE VAT = %(VAT)s;
            """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")

            cur.execute("""
                SELECT a.*,
                    CASE WHEN c.VAT_doctor IS NOT NULL THEN TRUE ELSE FALSE END AS is_in_consultation
                FROM appointment AS a
                LEFT JOIN consultation AS c
                    ON a.VAT_doctor = c.VAT_doctor AND a.date_timestamp = c.date_timestamp
                WHERE a.VAT_client = %(VAT)s
                ORDER BY a.date_timestamp DESC;
            """, {"VAT": VAT})
            appointments = cur.fetchall()
            app.logger.debug(f"Found {cur.rowcount} appointment(s).")

            cur.execute("""
                SELECT c.VAT_doctor,  c.date_timestamp, c.soap_s, c.soap_o, c.soap_a, c.soap_p
                FROM consultation AS c
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s
                ORDER BY c.date_timestamp;
            """, {"VAT": VAT})
            consultations = cur.fetchall()
            app.logger.debug(f"Found {cur.rowcount} consultation(s).")


    return render_template("clients/client_vat.html", client = client, consultations = consultations, appointments = appointments)
```

Screenshot of Web page





Description

Web page to display existing appointments and consultations for a specific client in the database.

For each displayed appointment, we included two options. One of them is the possibility of accessing a page to view/edit the appointment details: the button "View" redirects to "Update Appointment/Consultation" page. The other option is to create a consultation corresponding to the appointment: the button "Create" is only available if that corresponding consultation does not already exist. This button redirects to "Add Consultation".

For each displayed consultation there is the possibility to access a page to view/edit the consultation details: the button "View" redirects to "Update Appointment/Consultation" page.

This page also includes a button "Back" which redirects to "Clients".

Add Consultation

clients/create_consultation.html

In [ ]:
```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Create Consultation for client {{ client.name }} with VAT {{ client.vat }}{% endblock %}</h1>
{% endblock %}

{% block content %}
    <h2>Consultation</h2>

    <form id="createConsultationForm" method="post"
            action="/client/{{ client.vat }}/{{ appointment.vat_doctor }}/{{ appointment.date_timestamp }}/create_consultation2">
        <p>VAT Doctor: {{ appointment.vat_doctor }}</p>
        <p>Date Timestamp: {{ appointment.date_timestamp }}</p>

        <table>
            <thead>
                <tr>
                    <th>SOAP S</th>
                    <th>SOAP O</th>
```

```
                    <th>SOAP A</th>
                    <th>SOAP P</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>
                        <input type="text" name="soap_s" id="soap_s" value="{{ consultation.soap_s }}" maxlength="65535" required>
                    </td>
                    <td>
                        <input type="text" name="soap_o" id="soap_o" value="{{ consultation.soap_o }}" maxlength="65535" required>
                    </td>
                    <td>
                        <input type="text" name="soap_a" id="soap_a" value="{{ consultation.soap_a }}" maxlength="65535" required>
                    </td>
                    <td>
                        <input type="text" name="soap_p" id="soap_p" value="{{ consultation.soap_p }}" maxlength="65535" required>
                    </td>
                    <td>
                        <button type="submit">Create Consultation</button>
                    </td>
                </tr>
            </tbody>
        </table>
    </form>

    <!-- Button to redirect -->
    <form method="get" action="/client/{{ client.vat }}/{{ appointment.vat_doctor }}/{{ appointment.date_timestamp }}">
        <button type="submit">Back</button>
    </form>
{% endblock %}
```

Associated endpoints

In [ ]:

```
@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/create_consultation", methods=["GET"])
def add_consultation_dashboard(VAT, VAT_doctor, date_timestamp):
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
                FROM client
                WHERE VAT = %(VAT)s;
            """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")

            cur.execute("""
                SELECT c.VAT_doctor,  c.date_timestamp, c.soap_s, c.soap_o, c.soap_a, c.soap_p
                FROM consultation AS c
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s AND a.VAT_doctor = %(VAT_doctor)s AND a.date_timestamp = %(date_timestamp)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            consultation = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} consultation(s).")

            cur.execute("""
                SELECT *
                FROM appointment
                WHERE VAT_client = %(VAT)s AND VAT_doctor = %(VAT_doctor)s AND date_timestamp = %(date_timestamp)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            appointment = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} appointment(s).")

    return render_template("clients/create_consultation.html", client = client, consultation = consultation, appointment = appointment)

@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/create_consultation2", methods=["POST"])
def add_consultation2(VAT, VAT_doctor, date_timestamp):
    soap_s = request.form.get("soap_s")
    soap_o = request.form.get("soap_o")
    soap_a = request.form.get("soap_a")
    soap_p = request.form.get("soap_p")

    error = ""

    if error != "":
        flash(error)
        return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp + '/update_consultation')

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                INSERT INTO consultation (VAT_doctor, date_timestamp, SOAP_S, SOAP_O, SOAP_A, SOAP_P)
                VALUES
                (%(VAT_doctor)s, %(date_timestamp)s, %(soap_s)s, %(soap_o)s, %(soap_a)s, %(soap_p)s);
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp,
                    "soap_s": soap_s, "soap_o": soap_o, "soap_a": soap_a, "soap_p": soap_p})

            conn.commit()

    flash('Consultation created successfully.')

    return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)
```

Screenshot of Web page

Description

Form to create a new consultation corresponding to an existing appointment.

It is protected against:

- Empty soap_s;
- Empty soap_o;
- Empty soap_a;
- Empty soap_p;
- soap_s length higher than 65535 characters;
- soap_o length higher than 65535 characters;
- soap_a length higher than 65535 characters;
- soap_p length higher than 65535 characters.

All fields are required. "Back" redirects to "Appointment and Consultation" page. "Create Consultation" button redirects to "Update Appointment/Consultation". If an error occurs in this process, the page refreshes adding an error message to its display.

Update Appointment/Consultation

clients/consultation_desc.html

In [ ]:
```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Update Appointment/Consultation for client {{ client.name }} with VAT {{ client.vat }}{% endblock %}</h1>
{% endblock %}

{% block content %}

    <script>
        function redirectAddProcedure(VAT, doctorVAT, timestamp) {
            window.location.href = `/client/${VAT}/${doctorVAT}/${timestamp}/add_procedure`;
        }

        function redirectAddDiagnostic(VAT, doctorVAT, timestamp) {
            window.location.href = `/client/${VAT}/${doctorVAT}/${timestamp}/add_diagnostic`;
        }

        function redirectAddNurse(VAT, doctorVAT, timestamp) {
            window.location.href = `/client/${VAT}/${doctorVAT}/${timestamp}/add_nurse`;
        }
    </script>

    <button onclick="window.location.href='/client/{{ client.vat }}'">Back</button>

    <h2>Appointments</h2>
    <table>
        <thead>
            <tr>
                <th>VAT Doctor</th>
                <th>Date and Time</th>
                <th>VAT Client</th>
                <th>Description</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>{{ appointment.vat_doctor }}</td>
                <td>{{ appointment.date_timestamp }}</td>
                <td>{{ appointment.vat_client }}</td>
                <td>{{ appointment.description }}</td>
                <td>

                    <form action="/client/{{ client.vat }}/{{ appointment.vat_doctor }}/
```

```
                        {{ appointment.date_timestamp }}/update_appointment" method="post">
                    <button type="submit">Update Appointment</button>
                </form>
            </td>
        </tr>
    </tbody>
</table>

<h2>Consultation</h2>
<table>
    <thead>
        <tr>
            <th>VAT doctor</th>
            <th>Date and Time</th>
            <th>SOAP S</th>
            <th>SOAP O</th>
            <th>SOAP A</th>
            <th>SOAP P</th>
            <th>Action</th>
        </tr>
    </thead>
    {% if consultation != None %}
    <tbody>
        <tr>
            <td>{{ appointment.vat_doctor }}</td>
            <td>{{ appointment.date_timestamp }}</td>
            <td>{{ consultation.soap_s }}</td>
            <td>{{ consultation.soap_o }}</td>
            <td>{{ consultation.soap_a }}</td>
            <td>{{ consultation.soap_p }}</td>
            <td>

                <form action="/client/{{ client.vat }}/{{ appointment.vat_doctor }}
                        /{{ appointment.date_timestamp }}/update_consultation" method="post">
                    <button type="submit">Update Consultation</button>
                </form>
            </td>
        </tr>
    </tbody>
    {% endif %}
</table>

{% if consultation != None %}
<h2>Procedures</h2>
<table id="procedures_table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Description</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody>
        {% for procedure in procedures %}
            <tr>
                <td>{{ procedure.name }}</td>
                <td>{{ procedure.description }}</td>
                <td>
                    <form action="/client/{{ client.vat }}/{{ appointment.vat_doctor }}
                            /{{ appointment.date_timestamp }}/update_procedure/{{procedure.name}}" method="post">
                        <button type="submit">Update Procedure</button>
                    </form>

                    <form action="/client/{{ client.vat }}/{{ appointment.vat_doctor }}
                            /{{ appointment.date_timestamp }}/delete_procedure/{{procedure.name}}" method="post">
                        <button type="submit">Delete Procedure</button>
                    </form>
                </td>
            </tr>
        {% endfor %}
    </tbody>
</table>
<button type="button" onclick="redirectAddProcedure('{{ client.vat }}',
    '{{ appointment.vat_doctor }}', '{{ appointment.date_timestamp }}')">Add Procedure</button>

<h2>Diagnosis</h2>
<table id="diagnosis_table">
    <thead>
        <tr>
            <th>ID</th>
            <th>Description</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody>
        {% for diagnostic in diagnosis %}
            <tr>
                <td>{{ diagnostic.id }}</td>
                <td>{{ diagnostic.description }}</td>
                <td>
                    <form action="/client/{{ client.vat }}/{{ appointment.vat_doctor }}
                            /{{ appointment.date_timestamp }}/delete_diagnostic/{{diagnostic.id}}" method="post">
                        <button type="submit">Delete Diagnostic</button>
                    </form>
```

```
                    </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
    <button type="button" onclick="redirectAddDiagnostic('{{ client.vat }}',
        '{{ appointment.vat_doctor }}', '{{ appointment.date_timestamp }}')">Add Diagnostic</button>

    <h2>Nurses</h2>
    <table id="nurses_table">
        <thead>
            <tr>
                <th>VAT</th>
                <th>Name</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            {% for nurse in nurses %}
                <tr>
                    <td>{{ nurse.vat }}</td>
                    <td>{{ nurse.name }}</td>
                    <td>
                        <form action="/client/{{ client.vat }}/{{ appointment.vat_doctor }}
                                /{{ appointment.date_timestamp }}/delete_nurse/{{nurse.vat}}" method="post">
                            <button type="submit" >Delete Nurse</button>
                        </form>
                    </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>

    {% if nurses|length == 0 %}
        <button type="button" onclick="redirectAddNurse('{{ client.vat }}',
            '{{ appointment.vat_doctor }}', '{{ appointment.date_timestamp }}')">Add Nurse</button>
    {% endif %}
    {% endif %}

{% endblock %}
```

Associated endpoints

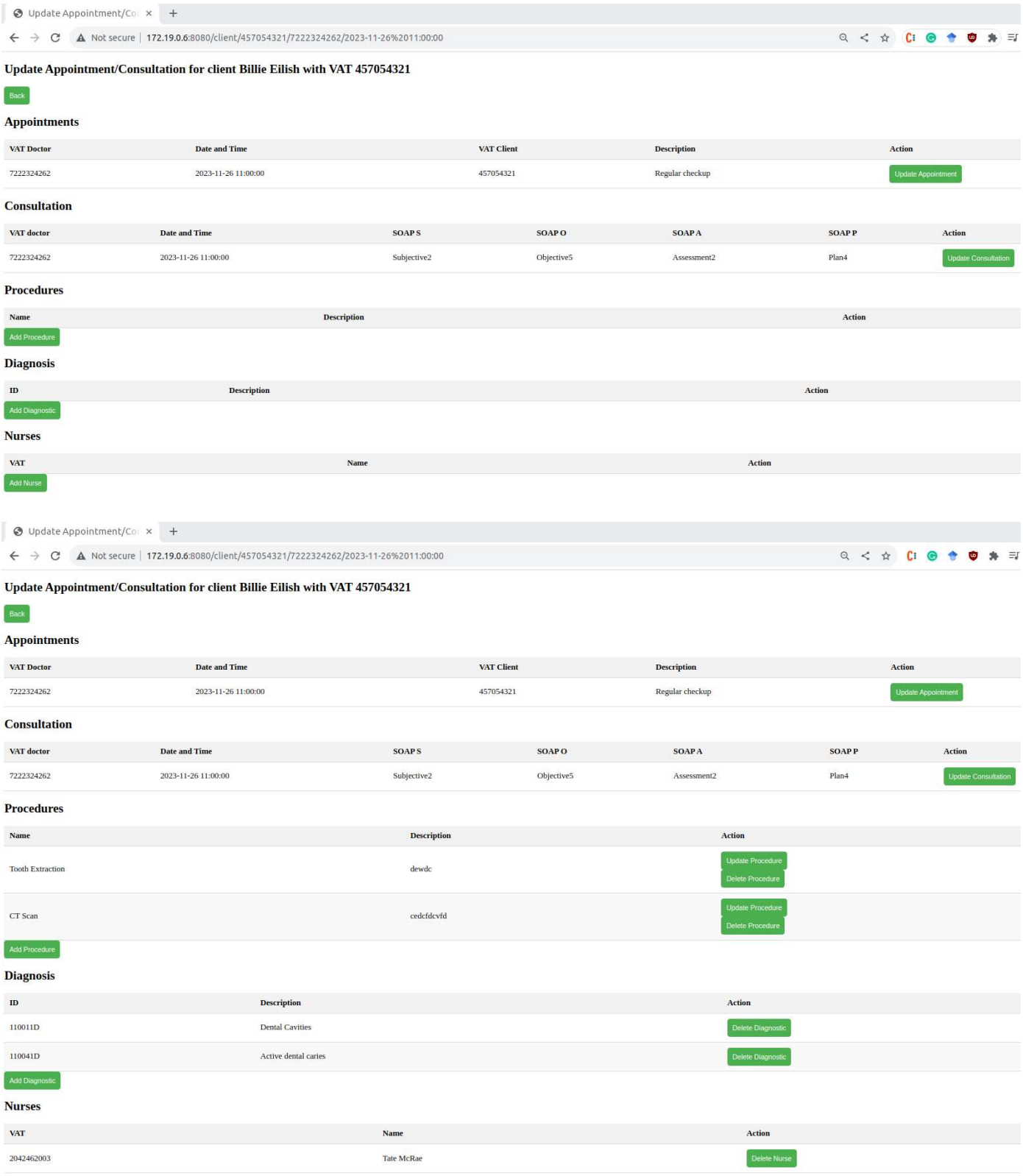
In [ ]:

```
@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>", methods=["GET"])
def consultation_desc(VAT, VAT_doctor, date_timestamp):

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
                FROM client
                WHERE VAT = %(VAT)s;
            """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")

            cur.execute("""
                SELECT *
                FROM appointment
                WHERE VAT_client = %(VAT)s AND VAT_doctor = %(VAT_doctor)s AND date_timestamp = %(date_timestamp)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            appointment = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} appointment(s).")

            cur.execute("""
                SELECT c.VAT_doctor,  c.date_timestamp, c.soap_s, c.soap_o, c.soap_a, c.soap_p
                FROM consultation AS c
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s AND a.VAT_doctor = %(VAT_doctor)s AND a.date_timestamp = %(date_timestamp)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            consultation = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} consultation(s).")

            cur.execute("""
                SELECT pc.name, pc.VAT_doctor, pc.date_timestamp, pc.description
                FROM procedure_in_consultation AS pc
                JOIN consultation AS c ON c.VAT_doctor = pc.VAT_doctor AND c.date_timestamp = pc.date_timestamp
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s AND a.VAT_doctor = %(VAT_doctor)s AND a.date_timestamp = %(date_timestamp)s
                ORDER BY pc.date_timestamp;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            procedures = cur.fetchall()
            app.logger.debug(f"Found {cur.rowcount} procedure(s).")

            cur.execute("""
                SELECT dc.ID, dc.description
                FROM diagnostic_code AS dc
                JOIN consultation_diagnostic AS cd ON cd.ID = dc.ID
                JOIN consultation AS c ON c.VAT_doctor = cd.VAT_doctor AND c.date_timestamp = cd.date_timestamp
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s AND a.VAT_doctor = %(VAT_doctor)s AND a.date_timestamp = %(date_timestamp)s
                ORDER BY cd.date_timestamp;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
```

```
            diagnosis = cur.fetchall()
            app.logger.debug(f"Found {cur.rowcount} diagnosi(s).")

            cur.execute("""
                SELECT n.VAT, e.name
                FROM nurse AS n
                JOIN employee AS e ON e.VAT = n.VAT
                JOIN consultation_assistant AS ca ON ca.VAT_nurse = n.VAT
                JOIN consultation AS c ON c.VAT_doctor = ca.VAT_doctor AND c.date_timestamp = ca.date_timestamp
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s AND a.VAT_doctor = %(VAT_doctor)s AND a.date_timestamp = %(date_timestamp)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            nurses = cur.fetchall()
            app.logger.debug(f"Found {cur.rowcount} nurse(s).")

    return render_template("clients/consultation_desc.html", client = client, consultation = consultation,
        appointment = appointment, procedures = procedures, diagnosis = diagnosis, nurses = nurses)

@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/delete_procedure/<name>", methods=["POST"])
def delete_procedure(VAT, VAT_doctor, date_timestamp, name):

    error = ""

    if error != "":
        flash(error)
        return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                DELETE FROM procedure_charting
                WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s AND name = %(name)s;
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "name": name})

            cur.execute("""
                DELETE FROM procedure_imaging
                WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s AND name = %(name)s;

            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "name": name})

            cur.execute("""
                DELETE FROM procedure_in_consultation
                WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s AND name = %(name)s;
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "name": name})

            conn.commit()

    flash('Procedure deleted successfully.')
    return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)

@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/delete_nurse/<VAT_nurse>", methods=["POST"])
def delete_nurse(VAT, VAT_doctor, date_timestamp, VAT_nurse):

    error = ""

    if error != "":
        flash(error)
        return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                DELETE FROM consultation_assistant
                WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s AND VAT_nurse = %(VAT_nurse)s;
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "VAT_nurse": VAT_nurse})

            conn.commit()

    flash('Nurse deleted successfully.')
    return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)

@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/delete_diagnostic/<ID>", methods=["POST"])
def delete_diagnostic(VAT, VAT_doctor, date_timestamp, ID):

    error = ""

    if error != "":
        flash(error)
        return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                DELETE FROM prescription
                WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s AND ID = %(ID)s;
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "ID": ID})

            cur.execute("""
                DELETE FROM consultation_diagnostic
                WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s AND ID = %(ID)s;
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "ID": ID})

            conn.commit()
```

```
    flash('Diagnostic deleted successfully.')
    return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)
```

Screenshot of Web page



Description

Web page to display an existing appointment and its corresponding consultation if it exists.

The page includes a button "Back" which redirects to "Appointments and Consultations".

For the appointment, there is an option to access a page to edit the appointment details: the button "Update Appointment" redirects to "Update Appointment" page.

The next described features are only on the page if the corresponding consultation exists.

For the consultation, there is an option to access a page to edit the consultation details: the button "Update Consultation" redirects to "Update Consultation" page.

For each consultation procedure, there are two options. One of them is to access a page to edit the procedure details: the button "Update Procedure" redirects to "Update Procedure" page. The other one is to delete the procedure from the consultation: the button "Delete Procedure". Then the page also includes a button "Add Procedure" which redirects to "Add Procedure" page.

For each consultation diagnostic, there is the option to delete the diagnostic from the consultation: the button "Delete Diagnostic". Then the page also includes a button "Add Diagnostic" which redirects to "Add Diagnostic" page.

For each consultation nurse, there is the option to delete the nurse from the consultation: the button "Delete Nurse". Then the page also includes a button "Add Nurse" which redirects to "Add Nurse" page. This add button is only displayed if there is no nurse associated with the consultation.

Update Appointment

clients/update_appointment.html

In [ ]:
```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Update Appointment for client {{ client.name }} with VAT {{ client.vat }}{% endblock %}</h1>
{% endblock %}

{% block content %}
    <h2>Appointment</h2>

    <form id="updateAppointmentForm" method="post" action="/client/{{ client.vat }}
            /{{ appointment.vat_doctor }}/{{ appointment.date_timestamp }}/update_appointment2">
        <p>VAT Doctor: {{ appointment.vat_doctor }}</p>
        <p>Date Timestamp: {{ appointment.date_timestamp }}</p>

        <table>
            <thead>
                <tr>
                    <th>Description</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>
                        <input type="text" name="description" id="description"
                            value="{{ appointment.description }}" maxlength="65535" required>
                    </td>
                    <td>
                        <button type="submit">Update Appointment</button>
                    </td>
```

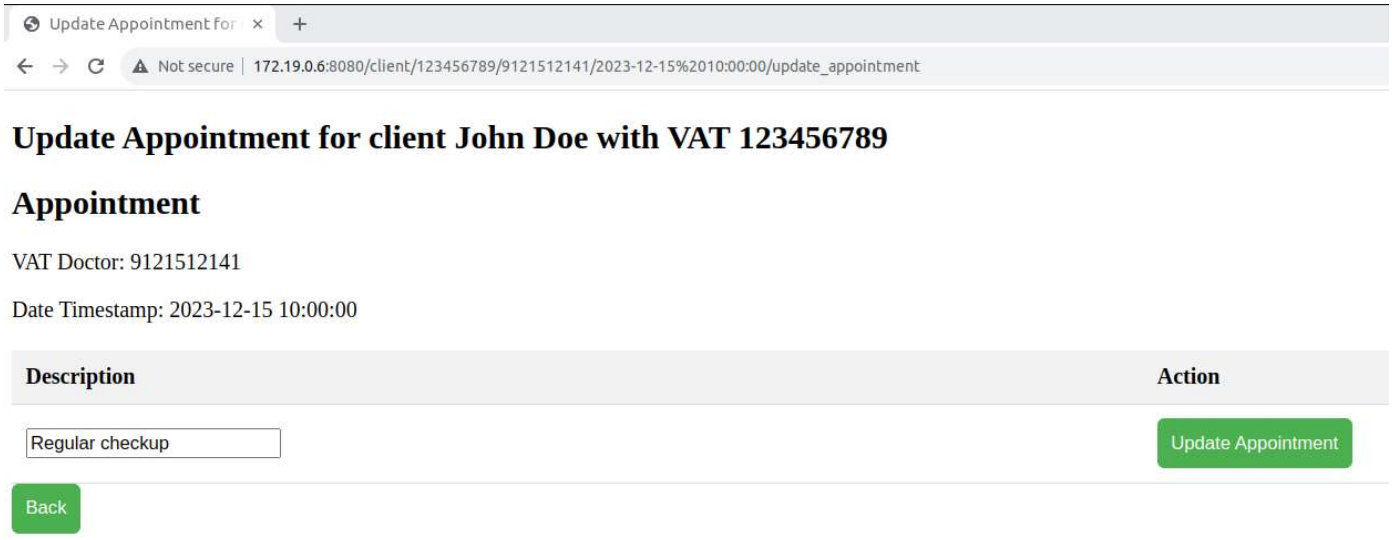
```
                </tr>
            </tbody>
        </table>
    </form>

    <!-- Button to redirect to "/client/<VAT>/<VAT_doctor>/<date_timestamp>" -->
    <button onclick="redirectToAppointment()">Back</button>

    <script>
        function redirectToAppointment() {
            // Construct the redirect URL
            var redirectUrl = "/client/{{ client.vat }}/{{ appointment.vat_doctor }}/{{ appointment.date_timestamp }}";

            // Perform the redirection
            window.location.href = redirectUrl;
        }
    </script>
{% endblock %}
```

Associated endpoints

In [ ]:

```
@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/update_appointment", methods=["POST"])
def update_appointment_dashboard(VAT, VAT_doctor, date_timestamp):
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
                FROM appointment
                WHERE VAT_client = %(VAT)s AND VAT_doctor = %(VAT_doctor)s AND date_timestamp = %(date_timestamp)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            appointment = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} appointment(s).")

            cur.execute("""
                    SELECT *
                    FROM client
                    WHERE VAT = %(VAT)s;
                """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")


    return render_template("clients/update_appointment.html", appointment = appointment, client = client)

@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/update_appointment2", methods=["POST"])
def update_appointment(VAT, VAT_doctor, date_timestamp):
    description = request.form.get("description")

    error = ""

    if error != "":
        flash(error)
        return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp + '/update_appointment')

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                UPDATE appointment
                SET description = %(description)s
                WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s;
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "description": description})

            conn.commit()

    flash('Appointment updated successfully.')
    return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)
```

Screenshot of Web page



Description

Form to update an appointment.

It is protected against a description length higher than 65535 characters and an empty description.

The description field is required. "Back" redirects to "Update Appointment/Consultation" page. "Update Appointment" button updates the appointment in the database and redirects to "Update Appointment/Consultation". If an error occurs in this process, the page refreshes adding an error message to its display.

Update Consultation

clients/update_consultation.html

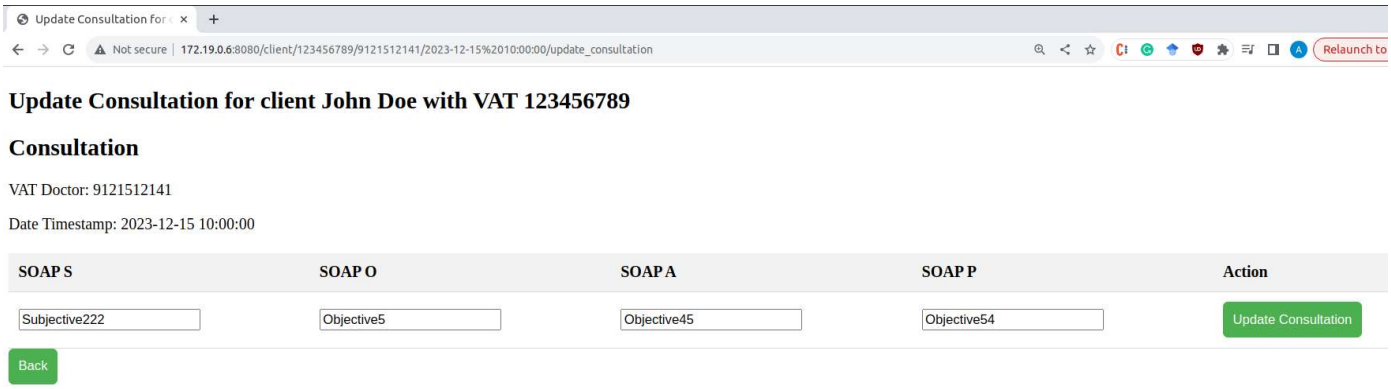
In [ ]:

```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Update Consultation for client {{ client.name }} with VAT {{ client.vat }}{% endblock %}</h1>
{% endblock %}

{% block content %}
    <h2>Consultation</h2>

    <form id="updateConsultationForm" method="post" action="/client/{{ client.vat }}
            /{{ consultation.vat_doctor }}/{{ consultation.date_timestamp }}/update_consultation2">
        <p>VAT Doctor: {{ consultation.vat_doctor }}</p>
        <p>Date Timestamp: {{ consultation.date_timestamp }}</p>

        <table>
            <thead>
                <tr>
                    <th>SOAP S</th>
                    <th>SOAP O</th>
                    <th>SOAP A</th>
                    <th>SOAP P</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>
                        <input type="text" name="soap_s" id="soap_s" value="{{ consultation.soap_s }}" maxlength="65535" required>
                    </td>
                    <td>
                        <input type="text" name="soap_o" id="soap_o" value="{{ consultation.soap_o }}" maxlength="65535" required>
                    </td>
                    <td>
                        <input type="text" name="soap_a" id="soap_a" value="{{ consultation.soap_a }}" maxlength="65535" required>
                    </td>
                    <td>
                        <input type="text" name="soap_p" id="soap_p" value="{{ consultation.soap_p }}" maxlength="65535" required>
                    </td>
                    <td>
                        <button type="submit">Update Consultation</button>
                    </td>
                </tr>
            </tbody>
        </table>
    </form>

    <!-- Button to redirect -->
    <form method="get" action="/client/{{ client.vat }}/{{ consultation.vat_doctor }}/{{ consultation.date_timestamp }}">
        <button type="submit">Back</button>
    </form>
{% endblock %}
```

Associated endpoints

In [ ]:
```python
@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/update_consultation", methods=["POST"])
def update_consultation_dashboard(VAT, VAT_doctor, date_timestamp):
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT c.VAT_doctor,  c.date_timestamp, c.soap_s, c.soap_o, c.soap_a, c.soap_p
                FROM consultation AS c
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s AND a.VAT_doctor = %(VAT_doctor)s AND a.date_timestamp = %(date_timestamp)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            consultation = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} consultation(s).")

            cur.execute("""
                SELECT *
                FROM client
                WHERE VAT = %(VAT)s;
            """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")

    return render_template("clients/update_consultation.html", consultation = consultation, client = client)

@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/update_consultation2", methods=["POST"])
def update_consultation(VAT, VAT_doctor, date_timestamp):
    soap_s = request.form.get("soap_s")
    soap_o = request.form.get("soap_o")
    soap_a = request.form.get("soap_a")
    soap_p = request.form.get("soap_p")

    error = ""

    if error != "":
        flash(error)
        return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp + '/update_consultation')

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                UPDATE consultation
                SET SOAP_S = %(soap_s)s, SOAP_O = %(soap_o)s, SOAP_A = %(soap_a)s, SOAP_P = %(soap_p)s
                WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s;
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp,
                    "soap_s": soap_s, "soap_o": soap_o, "soap_a": soap_a, "soap_p": soap_p})

            conn.commit()

    flash('Consultation updated successfully.')


    return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)
```

Screenshot of Web page



Description

Form to update a consultation.

It is protected against:

- Empty soap_s;
- Empty soap_o;
- Empty soap_a;
- Empty soap_p;
- soap_s length higher than 65535 characters;
- soap_o length higher than 65535 characters;
- soap_a length higher than 65535 characters;
- soap_p length higher than 65535 characters.

All fields are required. "Back" redirects to "Update Appointment/Consultation" page. "Update Consultation" button updates the consultation in the database and redirects to "Update Appointment/Consultation". If an error occurs in this process, the page refreshes adding an error message to its display.

Update Procedure

clients/update_procedure.html

In [ ]:
```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Update Procedure {{procedure.name}} for client {{ client.name }} with VAT {{ client.vat }}{% endblock %}</h1>
{% endblock %}

{% block content %}
    <h2>Procedure</h2>

    <form id="updateProcedureForm" method="post" action="/client/{{ client.vat }}
            /{{ procedure.vat_doctor }}/{{ procedure.date_timestamp }}/update_procedure2/{{ procedure.name }}">
        <p>VAT Doctor: {{ procedure.vat_doctor }}</p>
        <p>Date Timestamp: {{ procedure.date_timestamp }}</p>
        <p>Procedure name: {{ procedure.name }}</p>

        <table>
            <thead>
                <tr>
                    <th>Description</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>
                        <input type="text" name="description" id="description"
                            value="{{ procedure.description }}" maxlength="65535" required>
                    </td>
                    <td>
                        <button type="submit">Update Procedure</button>
                    </td>
                </tr>
            </tbody>
        </table>
    </form>

    <!-- Button to redirect -->
    <button onclick="window.location.href='/client/{{ client.vat }}
        /{{ procedure.vat_doctor }}/{{ procedure.date_timestamp }}'">Back</button>
{% endblock %}
```

Associated endpoints

In [ ]:
```
@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/update_procedure/<name>", methods=["POST"])
def update_procedure_dashboard(VAT, VAT_doctor, date_timestamp, name):
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT pc.name, pc.VAT_doctor, pc.date_timestamp, pc.description
                FROM procedure_in_consultation AS pc
                JOIN consultation AS c ON c.VAT_doctor = pc.VAT_doctor AND c.date_timestamp = pc.date_timestamp
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s AND a.VAT_doctor = %(VAT_doctor)s AND a.date_timestamp = %(date_timestamp)s
                    AND pc.name = %(name)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "name": name})
            procedure = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} procedure(s).")

            cur.execute("""
                SELECT *
                FROM client
                WHERE VAT = %(VAT)s;
            """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")

    return render_template("clients/update_procedure.html", procedure = procedure, client = client)

@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/update_procedure2/<name>", methods=["POST"])
def update_procedure(VAT, VAT_doctor, date_timestamp, name):
    description = request.form.get("description")

    error = ""

    if error != "":
        flash(error)
        return redirect('/client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp + '/update_procedure' + '/' + name)

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                UPDATE procedure_in_consultation
                SET description = %(description)s
```

```
            WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s AND name = %(name)s;
        """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "description": description, "name": name})

        conn.commit()

    flash('Procedure updated successfully.')
    return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)
```

Screenshot of Web page

Description

Form to update a procedure.

It is protected against a description length higher than 65535 characters and an empty description.

The description field is required. "Back" redirects to "Update Appointment/Consultation" page. "Update Procedure" button updates the procedure in the database and redirects to "Update Appointment/Consultation". If an error occurs in this process, the page refreshes adding an error message to its display.

Add Procedure

clients/add_procedure.html

In [ ]:

```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Add Procedure for client {{ client.name }} with VAT {{ client.vat }}{% endblock %}</h1>
{% endblock %}

{% block content %}
    <h2>Procedure</h2>

    <form id="addProcedureForm" method="post" action="/client/{{ client.vat }}
            /{{ consultation.vat_doctor }}/{{ consultation.date_timestamp }}/add_procedure2">
        <p>VAT Doctor: {{ consultation.vat_doctor }}</p>
        <p>Date Timestamp: {{ consultation.date_timestamp }}</p>

        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Description</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>
                        <select name="name">
                            {% for name in procedures_names %}
                                <option value="{{ name }}" {% if name == selected_name %}selected{% endif %}>{{ name }}</option>
                            {% endfor %}
                        </select>
                    </td>
                    <td>
                        <input type="text" name="description" value="{{ description }}" maxlength="65535" required>
                    </td>
                    <td>
                        <button type="submit">Add Procedure</button>
                    </td>
                </tr>
            </tbody>
```

```
        </table>
    </form>

    <!-- Button to redirect -->
    <button onclick="window.location.href='/client/{{ client.vat }}
        /{{ consultation.vat_doctor }}/{{ consultation.date_timestamp }}'">Back</button>
{% endblock %}
```

Associated endpoints

In [ ]:
```
@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/add_procedure", methods=["GET"])
def add_procedure_dashboard(VAT, VAT_doctor, date_timestamp):
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
                FROM client
                WHERE VAT = %(VAT)s;
            """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")

            cur.execute("""
                SELECT c.VAT_doctor,  c.date_timestamp, c.soap_s, c.soap_o, c.soap_a, c.soap_p
                FROM consultation AS c
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s AND a.VAT_doctor = %(VAT_doctor)s AND a.date_timestamp = %(date_timestamp)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            consultation = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} consultation(s).")

            cur.execute("""
                SELECT name
                FROM procedure_in_consultation;
                """)
            db_procedures = cur.fetchall()
            db_procedures = [row.name for row in db_procedures]
            app.logger.debug(f"Found {cur.rowcount} db_procedure(s).")

    return render_template("clients/add_procedure.html", client = client,
        consultation = consultation, procedures_names = db_procedures)


@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/add_procedure2", methods=["POST"])
def add_procedure(VAT, VAT_doctor, date_timestamp):
    description = request.form.get("description")

    error = ""

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT name
                FROM procedure;
                """)
            db_procedures = cur.fetchall()
            db_procedures = [row.name for row in db_procedures]
            app.logger.debug(f"Found {cur.rowcount} db_procedure(s).")

            cur.execute("""
                SELECT name
                FROM procedure_in_consultation
                WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s;
                """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            db_names = cur.fetchall()
            db_names = [row.name for row in db_names]
            app.logger.debug(f"Found {cur.rowcount} db_procedure(s).")

    name = request.form.get("name")

    if name not in(db_procedures) or name in(db_names):
        error = "Invalid procedure name"

    if error != "":
        flash(error)
        return redirect('/client/' + VAT + '/' + VAT_doctor  + '/' + date_timestamp + '/add_procedure')


    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                INSERT INTO procedure_in_consultation (name, VAT_doctor, date_timestamp, description)
                VALUES
                (%(name)s, %(VAT_doctor)s, %(date_timestamp)s, %(description)s);
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "name": name, "description": description})

            conn.commit()

    flash('Procedure created successfully.')
    return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)
```

Screenshot of Web page

Description

Form to add a procedure to a consultation.

It is protected against:

- description length higher than 65535 characters and empty description;
- invalid procedure names.

All fields are required. "Back" redirects to "Update Appointment/Consultation" page. "Add Procedure" button adds the procedure to the consultation in the database and redirects to "Update Appointment/Consultation". If an error occurs in this process, the page refreshes adding an error message to its display.

Add Diagnostic

clients/add_diagnostic.html

In [ ]:
```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Add Diagnostic for client {{ client.name }} with VAT {{ client.vat }} {% endblock %}</h1>
{% endblock %}

{% block content %}
    <h2>Diagnostic</h2>

    <form id="addDiagnosticForm" method="post" action="/client/{{ client.vat }}
            /{{ consultation.vat_doctor }}/{{ consultation.date_timestamp }}/add_diagnostic2">
        <p>VAT Doctor: {{ consultation.vat_doctor }}</p>
        <p>Date Timestamp: {{ consultation.date_timestamp }}</p>

        <table>
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>
                        <select name="ID">
                            {% for ID in IDs %}
                                <option value="{{ ID }}" {% if ID == selected_ID %}selected{% endif %}>{{ ID }}</option>
                            {% endfor %}
                        </select>
                    </td>
                    <td>
                        <button type="submit">Add Diagnostic</button>
                    </td>
                </tr>
            </tbody>
        </table>
    </form>

    <!-- Button to redirect -->
    <button onclick="window.location.href='/client/{{ client.vat }}
        /{{ consultation.vat_doctor }}/{{ consultation.date_timestamp }}'">Back</button>
{% endblock %}
```

Associated endpoints

In [ ]:
```
@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/add_diagnostic", methods=["GET"])
def add_diagnostic(VAT, VAT_doctor, date_timestamp):
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
```

```
                FROM client
                WHERE VAT = %(VAT)s;
            """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")

            cur.execute("""
                SELECT c.VAT_doctor,  c.date_timestamp, c.soap_s, c.soap_o, c.soap_a, c.soap_p
                FROM consultation AS c
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s AND a.VAT_doctor = %(VAT_doctor)s AND a.date_timestamp = %(date_timestamp)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            consultation = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} consultation(s).")

            cur.execute("""
                SELECT ID
                FROM diagnostic_code;
                """)
            db_ID = cur.fetchall()
            db_ID = [row.id for row in db_ID]
            app.logger.debug(f"Found {cur.rowcount} db_ID(s).")


    return render_template("clients/add_diagnostic.html", client = client, consultation = consultation, IDs = db_ID)

@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/add_diagnostic2", methods=["POST"])
def add_diagnostic2(VAT, VAT_doctor, date_timestamp):

    error = ""
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT ID
                FROM diagnostic_code;
                """)
            db_ID = cur.fetchall()
            db_ID = [row.id for row in db_ID]
            app.logger.debug(f"Found {cur.rowcount} db_ID(s).")

            cur.execute("""
                SELECT ID
                FROM consultation_diagnostic
                WHERE date_timestamp = %(date_timestamp)s AND VAT_doctor = %(VAT_doctor)s;               ;
                """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            db_ID2 = cur.fetchall()
            db_ID2 = [row.id for row in db_ID2]
            app.logger.debug(f"Found {cur.rowcount} db_ID2(s).")

    ID = request.form.get("ID")

    if ID not in(db_ID) or ID in(db_ID2):
        error = "Invalid ID"

    if error != "":
        flash(error)
        return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp + '/add_diagnostic')

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                INSERT INTO consultation_diagnostic (VAT_doctor, date_timestamp, ID)
                VALUES
                (%(VAT_doctor)s, %(date_timestamp)s, %(ID)s);
            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "ID": ID})

            conn.commit()

    flash('Diagnostic created successfully.')
    return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)
```

Screenshot of Web page

Description

Form to add a diagnostic to a consultation.

It is protected against invalid procedure ID.

The ID field is required. "Back" redirects to "Update Appointment/Consultation" page. "Add Diagnostic" button adds the diagnostics to the consultation in the database and redirects to "Update Appointment/Consultation". If an error occurs in this process, the page refreshes adding an error message to its display.

Add Nurse

clients/add_nurse.html

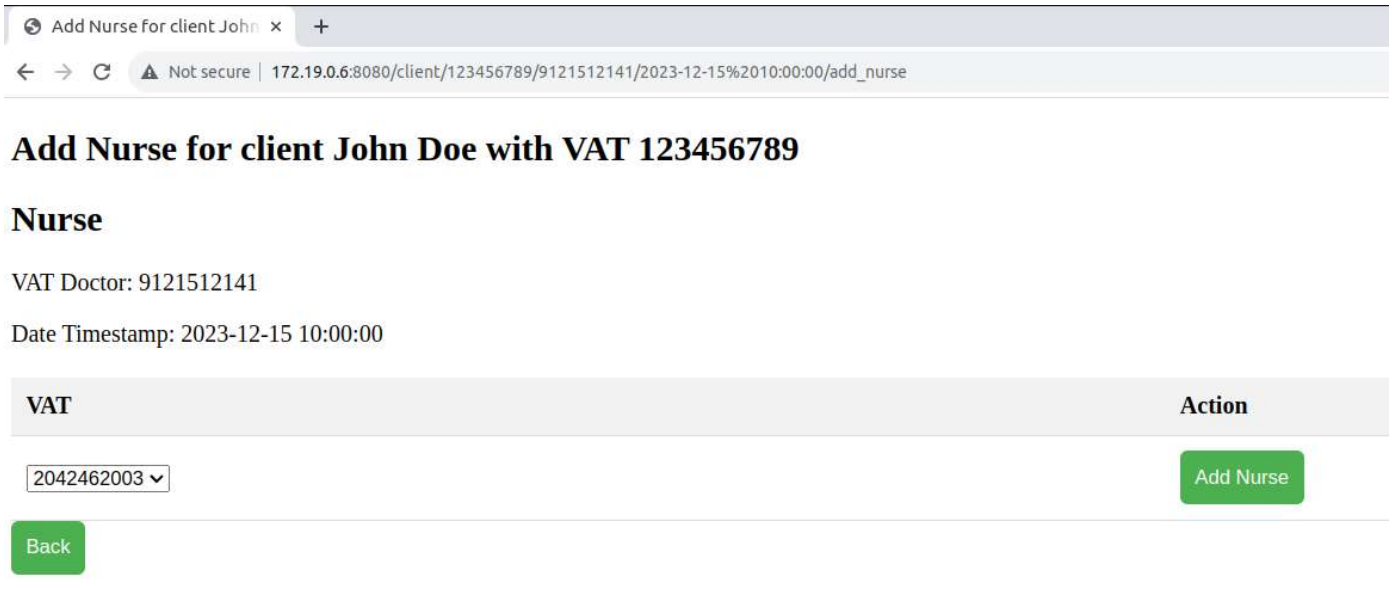
In [ ]:
```
{% extends 'base.html' %}

{% block header %}
    <h1>{% block title %}Add Nurse for client {{ client.name }} with VAT {{ client.vat }} {% endblock %}</h1>
{% endblock %}

{% block content %}
    <h2>Nurse</h2>

    <form id="addNurseForm" action="/client/{{ client.vat }}/{{ consultation.vat_doctor }}
            /{{ consultation.date_timestamp }}/add_nurse2" method="post">
        <p>VAT Doctor: {{ consultation.vat_doctor }}</p>
        <p>Date Timestamp: {{ consultation.date_timestamp }}</p>

        <table>
            <thead>
                <tr>
                    <th>VAT</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>
                        <select name="VAT">
                            {% for nurse_VAT in VAT_nurses %}
                                <option value="{{ nurse_VAT }}"
                                    {% if nurse_VAT == selected_VAT %}selected{% endif %}>{{ nurse_VAT }}</option>
                            {% endfor %}
                        </select>
                    </td>

                    <td>
                        <button type="submit">Add Nurse</button>
                    </td>
                </tr>
            </tbody>
        </table>
    </form>

    <!-- Button to redirect -->
    <button onclick="window.location.href='/client/{{ client.vat }}
        /{{ consultation.vat_doctor }}/{{ consultation.date_timestamp }}'">Back</button>
{% endblock %}
```

Associated endpoints

In [ ]:
```
@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/add_nurse", methods=["GET"])
def add_nurse_dashboard(VAT, VAT_doctor, date_timestamp):
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT *
```

```
                FROM client
                WHERE VAT = %(VAT)s;
            """, {"VAT": VAT})
            client = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} client(s).")

            cur.execute("""
                SELECT c.VAT_doctor,  c.date_timestamp, c.soap_s, c.soap_o, c.soap_a, c.soap_p
                FROM consultation AS c
                JOIN appointment AS a ON c.VAT_doctor = a.VAT_doctor AND c.date_timestamp = a.date_timestamp
                WHERE a.VAT_client = %(VAT)s AND a.VAT_doctor = %(VAT_doctor)s AND a.date_timestamp = %(date_timestamp)s;
            """, {"VAT": VAT, "VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp})
            consultation = cur.fetchone()
            app.logger.debug(f"Found {cur.rowcount} consultation(s).")

            cur.execute("""
                SELECT VAT
                FROM nurse;
                """)
            db_VAT_nurses = cur.fetchall()
            db_VAT_nurses = [row.vat for row in db_VAT_nurses]
            app.logger.debug(f"Found {cur.rowcount} db_VAT_nurses.")


    return render_template("clients/add_nurse.html", client = client, consultation = consultation, VAT_nurses = db_VAT_nurses)

@app.route("/client/<VAT>/<VAT_doctor>/<date_timestamp>/add_nurse2", methods=["POST"])
def add_nurse(VAT, VAT_doctor, date_timestamp):

    error = ""
    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                SELECT VAT
                FROM nurse;
                """)
            db_VAT_nurses = cur.fetchall()
            db_VAT_nurses = [row.vat for row in db_VAT_nurses]
            app.logger.debug(f"Found {cur.rowcount} db_VAT_nurses.")

    VAT_nurse = request.form.get("VAT")

    if VAT_nurse not in(db_VAT_nurses):
        error = "Invalid VAT_nurse"

    if error != "":
        flash(error)

        return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp + '/add_nurse')

    with psycopg.connect(conninfo=DATABASE_URL) as conn:
        with conn.cursor(row_factory=namedtuple_row) as cur:
            cur.execute("""
                INSERT INTO consultation_assistant(VAT_doctor, date_timestamp, VAT_nurse)
                VALUES
                (%(VAT_doctor)s, %(date_timestamp)s, %(VAT_nurse)s);

            """, {"VAT_doctor": VAT_doctor, "date_timestamp": date_timestamp, "VAT_nurse": VAT_nurse})

            conn.commit()

    flash('Nurse created successfully.')
    return redirect('/' + 'client' + '/' + VAT + '/' + VAT_doctor + '/' + date_timestamp)
```

Screenshot of Web page



Description

Form to add a nurse to a consultation.

It is protected against invalid nurse's VAT.

The VAT field is required. "Back" redirects to "Update Appointment/Consultation" page. "Add Nurse" button adds the nurse to the consultation in the database and redirects to "Update Appointment/Consultation". If an error occurs in this process, the page refreshes adding an error message to its display.

Deploy

We performed the deploy of the app, using fly.io and PostgreSQL Server @ Tecnico, as suggested in https://github.com/bdist/app/wiki.

APP LINK: https://sibd-deploy-app-2.fly.dev/