Projeto de Base de Dados 2023/2024

André Lourenço Albuquerque nº2022231505
João Pedro Gonçalves Ventura Barata nº2022245160
Rodrigo José Morenito Borges nº2022244993

Manual de Instalação

Para começar , deverá instalar o PostgreSQL pelo site oficial e escolher o instalador compatível com o seu sistema operativo e seguir as instruções de instalação
De seguida deve dirigir-se ao site oficial do Python e fazer download do instalador mais recente da versão do Python , novamente de acordo com o seu sistema operativo e seguir as instruções de instalação.
Ao fim de instalar estas duas ferramentas , para interagir com a API , deve instalar pelo site oficial , o Postman , que é uma ferramenta conhecida por fazer o teste e o desenvolvimento de APIs de forma acessível , sem esquecer de verificar a documentação de modo a perceber como são enviados pedidos por HTTP.

SETUP

PGADMIN

Criar a base de dados

Primeiramente deve iniciar sessão no utilizador postgres ou em um utilizador que tenha permissões , e após isso tem de criar uma base de dados.

Criar as tabelas

Ao fim de ter a base de dados criada e o respetivo utilizador deve executar o ficheiro dos triggers e ficheiro que contém as tabelas que foram geradas pelo Diagrama ER que desenvolvemos na meta 1 e ao longo do trabalho fomos adaptando às necessidades de cada funcionalidade.

Importação de bibliotecas

Em relação às bibliotecas , as que utilizamos foram o Flask e o Psycopg2.
-> pip install flask
-> pip install psycopg2
Deve posteriormente executar o ficheiro de python que lhe é fornecido (projeto.py)

Manual do Utilizador

Operações presentes na Base de dados:

Assistant
- See Appointments
- See Prescriptions
- Schedule Surgery

- List Top 3 patients
- Daily Summary
- Generate a monthly report

Patient

-Schedule Appointment
-See Appointments
-See Prescriptions
-Execute Payment

Doctor

-Check any patient's prescriptions
-Add Prescriptions

Nurse

-Check any patient's prescriptions

Register Patient

POST http://localhost:8080/dbproj/register/patient

BODY (json):{ "username": "andre", "password": "coimbra", "address": "Coimbra", "phone_number": "966234050"}

RESPONSE(json):{
"results": 1, (user_id)
"status": 200
}

Register Assistant

POST http://localhost:8080/dbproj/register/assistant

BODY (json):{
"username": "alex",
"password": "fafe",
"address": "Fafe",
"phone_number": "969507891"
}

RESPONSE(json):{
   "results": 2, (user_id)
   "status": 200
}

Register Nurse

POST http://localhost:8080/dbproj/register/nurse

BODY (json):{
   "username": "Carolina",
   "password": "viseu",
   "address": "Viseu",
   "phone_number": "0678001230",
   "category_name": "ola"
}

RESPONSE(json):{
   "results": 3, (user_id)
   "status": 200
}

Register Doctor

POST http://localhost:8080/dbproj/register/doctor

BODY (json):{
   "username": "Madeiras",
   "password": "azores",
   "address": "Azores",
   "phone_number": "1987554320",
   "licenseid": "19",
   "specializations": ["Cardiologia", "Neurologia"]
}

RESPONSE(json):{
   "results": 4, (user_id)
   "status": 200
}

User Authentication:

PUT **http://localhost:8080/dbproj/user**

BODY (json):{
"username": "andre",
"password": "coimbra"
}

RESPONSE(json):{ "results":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjpbMV19.Fi2shp6wQmmUHRbqQBkMzp00My0eS3Ab4UgirsKmHqM", (token)
"status": 200
}

Schedule Appointment (only patient)

POST **http://localhost:8080/dbproj/appointment**

AUTHORIZATION: Bearer Token ;
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjpbMV19.Fi2shp6wQmmUHRbqQBkMzp00My0eS3Ab4UgirsKmHqM (patient token)

BODY (json):{
"doctor_id": 4,
"date": "2024-06-01T10:00:00",
"nurse_ids": [3]
}

RESPONSE(json):{
"results": 1, (appointment_id)
"status": 200
}

See Appointments (only assistants or targeted patient)

GET **http://localhost:8080/dbproj/appointments/{patient_id}**

AUTHORIZATION: Bearer Token ;
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjpbMV19.Fi2shp6wQmmUHRbqQBkMzp00My0eS3Ab4UgirsKmHqM (targeted patient token) or (assistants)

RESPONSE(json):{

```
  "results": [
    {
      "date": "Sat, 01 Jun 2024 10:00:00 GMT",
      "doctor_id": 5,
      "doctor_name": "Madeiras",
      "id": 1,
      "nurses": [
        {
          "nurse_id": 3,
          "nurse_name": "Carolina"
        }
      ]
    }
  ],
  "status": 200
}
```

Schedule Surgery (only assistants)

POST http://127.0.0.1:8080/dbproj/surgery/
POST http://127.0.0.1:8080/dbproj/surgery/{hospitalization_id}

AUTHENTICATION(json): Bearer Token;
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjpbMl19.mYQkGlahGjXkkJZh9_UTKczoQ_mm1iSPoYG8ZgFDNdI (token assistant)

BODY(json):{"patient_id":1,"patient_user_data_id_user":1,"doctor_id":4,"nurses":[[3, "ola"],[5, "ola"]], "date": "2024-06-01T10:50:00","responsiblenurse": "Carolina","date_start": "2024-05-25T10:50:00","date_end":"2024-06-01T12:50:00","nurses_employees_user_data_id_user": 3, "assistants_employees_user_data_id_user": 2,"type":"cardiac"}

RESPONSE(json):{

```
  "results": {
    "date": "2024-06-01T10:50:00",
    "doctor_id": 4,
    "hospitalization_id": (id if provided) or null,
    "patient_id": 1,
    "surgery_id": 1,
    "type": "cardiac"
  },
  "status": 200
}
```

Get Prescriptions (only employees or targeted patient)

GET http://localhost:8080/dbproj/appointments/{person_id}

AUTHORIZATION: Bearer Token ;
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjpbMV19.Fi2shp6wQmmUHRbqQBkMzp00My0eS3Ab4UgirsKmHqM (targeted patient token) or (employees)

RESPONSE(json):{
"prescriptions": [],
"status": 200
}

or

RESPONSE(json):

```
{
    "prescriptions": [
        {
            "amount": 1,
            "duration": 3,
            "medicine_name": "brufen",
            "prescription_date": "2024-05-28 12:42:52",
            "prescription_id": 1,
            "validity": "2025-12-12"
        },
        {
            "amount": 3,
            "duration": 5,
            "medicine_name": "benuron",
            "prescription_date": "2024-05-28 12:42:52",
            "prescription_id": 1,
            "validity": "2025-12-12"
        }
    ],
    "status": 200
}
```

Add Prescriptions (only doctor)

AUTHENTICATION(json): Bearer Token;
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjpbNF19.yz8ioJtYLPwtz13iHGIBlh
DYxYHU725Y88cLQNAdfeo (token doctor)

POST **http://localhost:8080/dbproj/prescription/**
BODY (json):{ "evento": "hospitalization","event_id": 2,"validity": "2025-12-12","medicines": [
{"medicine": "brufen", "posology_dose": 1, "posology_frequency": 3 , "side_effects": [{
"name": "Nausea", "probability": 0.5, "severity": 2}] }] }

RESPONSE(json):{
  "results": 1,
  "status": 200
}

Execute Payment (only patient can pay his/her bills)

POST http://localhost:8080/dbproj/bills/{bill_id}

BODY (json):{ "amount": 900, "payment_method": "payper", "bill_type":"appointment"
}
RESPONSE(json):{ "results": 700 (amount remaining to pay), "status": 200 }

List Top 3 patients

GET
BODY (json):
RESPONSE(json):

Daily Summary

GET
BODY (json):
RESPONSE(json):{ "amount_spent": float, "prescriptions": int , surgeries:int }

Generate a monthly report (only assistants)

GET **http://localhost:8080/dbproj/report**

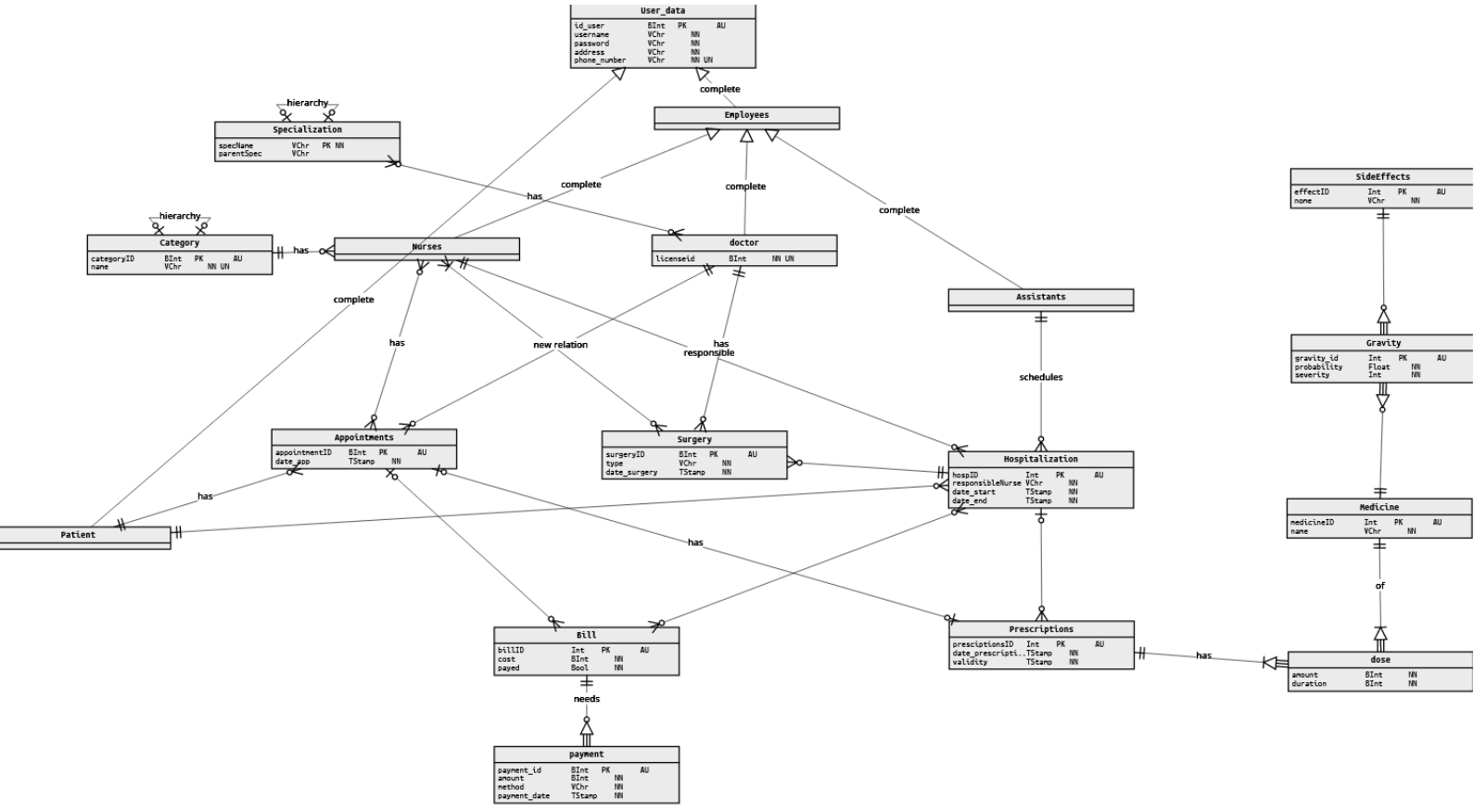
AUTHENTICATION (json): Bearer: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjpbMl19.mYQkGlahGjXkkJZh9_UTKczoQ_mm1iSPoYG8ZgFDNdI (token assistant)

RESPONSE(json):{
  "results": [
    {
      "doctor": "Madeiras",
      "month": "month_6",
      "surgeries": 1
    }
  ],
  "status": "success"
}

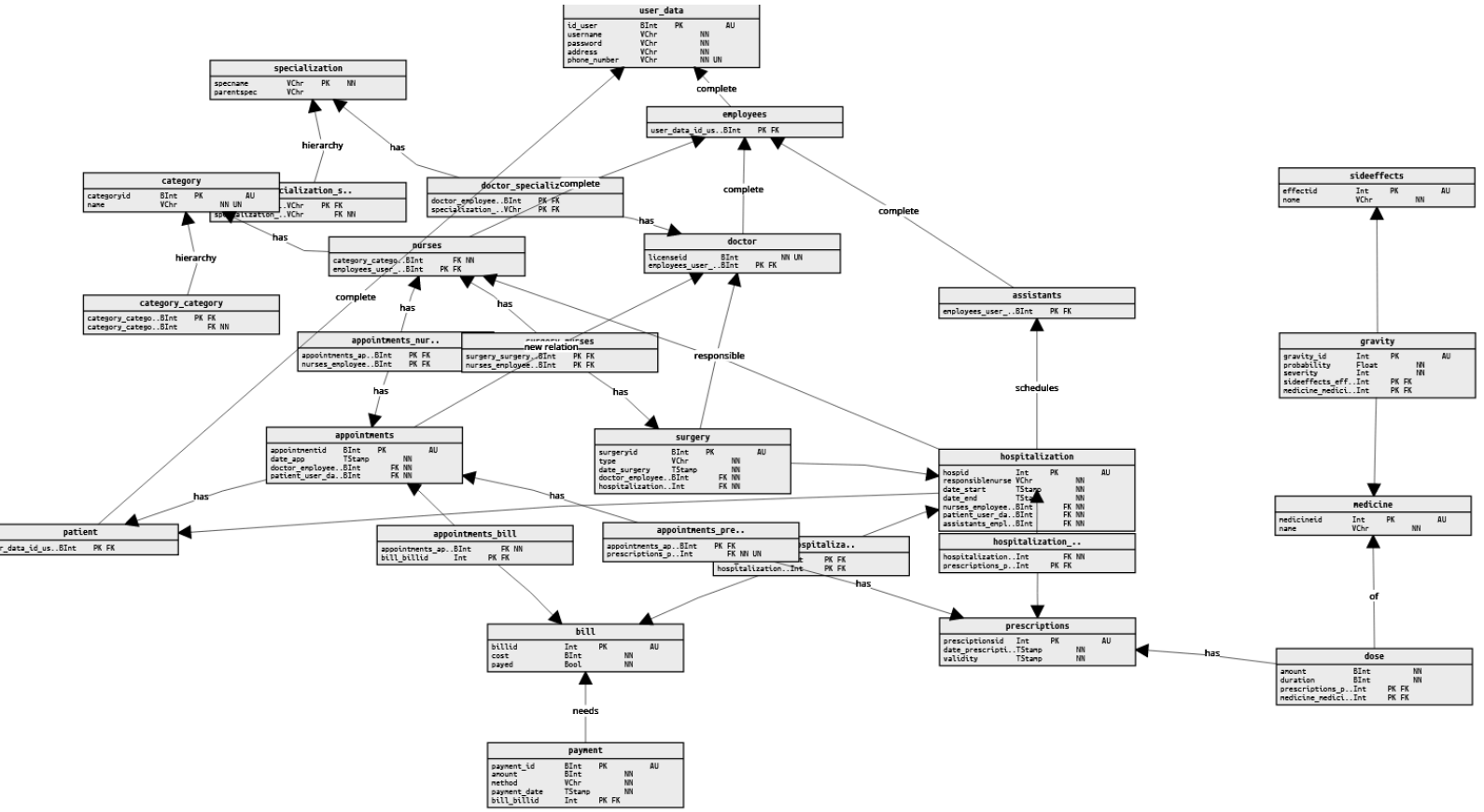
Trigger usados : trigger_create_bill_after_appointment , surgery_after_insert_trigger.

ER Final e Modelo de Dados Relacional

Diagrama ER Final



Modelo de Dados Relacional



Detalhes e informações adicionais relevantes

Além da implementação das funções , em alguns casos era necessário a utilização de triggers de modo a tornar o programa mais eficiente , foi o caso de criar uma bill após um appointment , onde o trigger_create_bill_after_appointment cria uma nova fatura para a consulta e a relaciona com a mesma e é chamado sempre que um utilizador faz a inserção de uma nova consulta. Já o surgery_after_insert_trigger é chamado após a inserção de uma surgery e se já existir uma bill , atualiza-a com o custo da surgery , senão cria uma bill nova.

Conclusão

Em suma , podemos concluir que o nosso projeto foi de encontro com aquilo que foi pedido no enunciado , que era uma aplicação com uma Base de Dados que pudesse guardar e processar todos os dados relacionados a esse hospital. Foi um trabalho que nos deu uma visão global de como funciona uma Base de Dados e como tudo se relaciona à volta. De realçar que tal como pedido usamos o PostgreSQL como DBMS , a ferramenta POSTMAN que permitiu enviar pedidos HTTP e na arquitetura da aplicação da base de dados incluímos uma REST API , que além de fazer com que haja comunicação com o servidor , dá nos a verificação de cada funcionalidade (status code).