



# Universidade Federal do Ceará

## **Relatório de Projeto de Banco de Dados**

Discentes: Tailan de Souza Oliveira (553381)

Kaio Davidson Santos Sampaio (554698)

Docente: Livia Almada Cruz

Disciplina: Fundamentos de Banco de Dados (FBD)

Título: Sistema de Gerenciamento de Remédios

07 de Abril de 2024

Nota ABNT: Este relatório segue as normas da Associação Brasileira de Normas Técnicas (ABNT).

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Requisitos</b>	<b>1</b>
2.1	Requisitos Funcionais . . . . .	2
2.2	Requisitos de Relatórios . . . . .	4
<b>3</b>	<b>Desenvolvimento</b>	<b>4</b>
3.1	Entidades, Atributos e Relacionamentos . . . . .	4
3.2	Entidade: Remédio . . . . .	4
3.3	Entidade: Fornecedor . . . . .	5
3.4	Entidade: Pedido . . . . .	6
3.5	Entidade: Cliente . . . . .	6
3.6	Entidade: Funcionário . . . . .	7
3.7	Entidade: Estoque . . . . .	7
<b>4</b>	<b>Diagrama ER/EER</b>	<b>8</b>
<b>5</b>	<b>Delegação de Atividades</b>	<b>8</b>
<b>6</b>	<b>Conclusão</b>	<b>9</b>
<b>7</b>	<b>Introdução</b>	<b>10</b>
<b>8</b>	<b>Estrutura do Banco de Dados</b>	<b>10</b>
8.1	Script para criação do banco de dados . . . . .	10
8.2	Populando as tabela de maneira não aleatória . . . . .	14
8.3	Ver todas tabelas com os dados informados; . . . . .	19
8.4	Logo abaixo modelo relacional do projeto de gerenciamento de remédios . . . . .	21
<b>9</b>	<b>O banco de dados é estruturado com as seguintes tabelas principais:</b>	<b>21</b>
9.1	Tabela <code>remedio</code> . . . . .	21
9.2	Tabela <code>genericos</code> . . . . .	21
9.3	Tabela <code>tarjas</code> . . . . .	22
9.4	Tabela <code>de_marcas</code> . . . . .	22
9.5	Tabela <code>pedido</code> . . . . .	22
9.6	Tabela <code>esta_incluido</code> . . . . .	23
9.7	Tabela <code>fornecedor</code> . . . . .	23
9.8	Tabela <code>solicitacao</code> . . . . .	23

9.9	Tabela <code>funcionario</code> . . . . .	24
9.10	Tabela <code>realiza</code> . . . . .	24
9.11	Tabela <code>estoque</code> . . . . .	24
9.12	Tabela <code>possui</code> . . . . .	25
<b>10</b>	<b>Descrição das Tabelas</b>	<b>25</b>
10.1	Tabela <code>cliente</code> . . . . .	25
10.2	Tabela <code>telefone</code> . . . . .	25
10.3	Tabela <code>historico_compra</code> . . . . .	26
10.4	Tabela <code>faz_parte</code> . . . . .	26
10.5	Tabela <code>tem</code> . . . . .	26
<b>11</b>	<b>API e Funcionalidades</b>	<b>27</b>
11.1	CRUD para Tabela Remédio . . . . .	27
11.2	CRUD para Tabela Fornecedor . . . . .	27
<b>12</b>	<b>Dado a criação do banco de dados e de ter populado suas tabelas de maneiras não aleatória, aqui está o resultado de maneira individual</b>	<b>27</b>
<b>13</b>	<b>Exemplos de Implementação</b>	<b>27</b>
13.1	Atributos Multivalorados . . . . .	27
13.2	Hierarquia de Herança . . . . .	28
13.3	Conclusão . . . . .	28
<b>14</b>	<b>Como Funciona um CRUD</b>	<b>29</b>
14.1	Imagem original sem nenhuma implementação (Tabela remédios)	29
14.2	Create (Criar) . . . . .	29
14.3	Após usar o CREATE . . . . .	32
14.4	Read (Ler) . . . . .	32
14.5	Resultando após usar o READ . . . . .	35
14.6	Antes de utilizar o UPDATE . . . . .	35
14.7	Update (Atualizar) . . . . .	35
14.8	Após usar o UPDATE . . . . .	39
14.9	Delete (Excluir) . . . . .	39
<b>15</b>	<b>Conclusão</b>	<b>40</b>
<b>16</b>	<b>Referências</b>	<b>43</b>
<b>17</b>	<b>Link do Repositório</b>	<b>43</b>

# 1 Introdução

Neste relatório, delineamos o projeto de banco de dados para um sistema abrangente de gerenciamento de medicamentos. Este sistema foi concebido com o objetivo primordial de aprimorar o controle e a gestão de informações essenciais relacionadas a clientes, produtos (remédios), vendas e funcionários, direcionado especificamente para farmácias ou estabelecimentos similares. Através da implementação deste sistema, buscamos oferecer uma solução eficiente para otimizar processos, garantindo um gerenciamento fluido e preciso das operações diárias, desde o controle de estoque até o acompanhamento das transações e interações com clientes e fornecedores.

## 2 Requisitos

A seguir, apresentamos os requisitos do sistema de gerenciamento de remédios:

- Entidades Principais
  1. Cliente
  2. Remédio
  3. Fornecedor
  4. Funcionário
  5. Estoque
  6. Pedido
- Relacionamentos
  1. Fornecedor-Pedido
  2. Funcionário-Pedido
  3. Remédio-Pedido
  4. Estoque-Remédio
  5. Remédio-Cliente
  6. Cliente-Histórico\_Compras
  7. Histórico\_Compras-Remédio
- Atributos
  1. Cliente:

- Nome, cpf, Telefone, Endereço, Número da Casa, Bairro, Cidade, Rua.
- 2. Remédio: Atributos e subgrupos
  - Genéricos, Tarjas, Cores da Tarjas, Outras Categorias, de Marca, nome, descrição, id
- 3. Fornecedor:
  - Nome de Empresa, Email, Endereço, Número, Rua, Bairro, Cidade, Estado, CNPJ.
- 4. Funcionário:
  - Salário, ID Funcionário, Cargo, Nome.
- 5. Estoque:
  - Quantidade no Estoque, ID\_estoque, Data de Entrada no Estoque, Data de validade, Unidade de Medida, Unidade, Cartelas, Caixas, Frascos
- 6. Pedido:
  - Data do Pedido, Quantidade de Remédios Solicitados, Status do Pedido, ID Pedido, Código de Rastreamento.
- Chaves Primárias e Estrangeiras
  - Consulte as chaves primárias e estrangeiras conforme mencionadas no documento original.
- Restrições de Integridade
  - Restrições de integridade referencial para garantir que as chaves estrangeiras estejam sempre relacionadas a uma chave primária existente.
  - Restrições de domínio para garantir que os valores dos atributos estejam dentro de um intervalo aceitável (por exemplo, preço não pode ser negativo).

## 2.1 Requisitos Funcionais

- **RF001 - Cadastro de Medicamentos:**
  - **Descrição:** Permitir o cadastro, edição, remoção e listagem de medicamentos. Cada medicamento deve ter um nome e descrição.
  - **Restrição:** O nome do medicamento deve ser único no sistema.

- **Subgrupos:**
  - \* **Genéricos:**
    - **Descrição:** Medicamentos genéricos produzidos por diferentes laboratórios.
  - \* **De Marca:**
    - **Descrição:** Medicamentos de marca registrada produzidos por empresas específicas.
- **RF002 - Registro de Fornecedores:**
  - **Descrição:** Possibilitar o registro de novos fornecedores, incluindo informações como nome da empresa, CNPJ, endereço e contato.
  - **Restrição:** O CNPJ do fornecedor deve ser único no sistema.
- **RF003 - Registros de Pedidos:**
  - **Descrição:** Permitir que os funcionários registrem os pedidos de medicamentos feitos pelos clientes, incluindo detalhes como a data do pedido, quantidade solicitada, status do pedido e cliente associado.
  - **Restrição:** Deve ser possível associar vários medicamentos a um único pedido.
- **RF004 - Controle de Estoque:**
  - **Descrição:** Manter um registro atualizado do estoque de medicamentos, incluindo a quantidade disponível, data de entrada, data de validade e unidade de medida.
  - **Restrição:** Deve ser possível visualizar o estoque disponível para cada medicamento.
- **RF005 - Registro de Clientes:**
  - **Descrição:** Permitir o registro de novos clientes, incluindo informações como nome, CPF, telefone, endereço e histórico de compras.
  - **Restrição:** O CPF do cliente deve ser único no sistema.
- **RF006 - Geração de Relatórios:**
  - **Descrição:** Gerar relatórios sobre vendas, estoque, clientes, fornecedores e pedidos para auxiliar na gestão do estabelecimento.

- **Restrição:** Os relatórios devem ser facilmente acessíveis e customizáveis.
- **RF007 - Autenticação de Usuários:**
  - **Descrição:** Implementar um sistema de autenticação de usuários para garantir que apenas funcionários autorizados possam acessar as funcionalidades do sistema.
  - **Restrição:** Cada usuário deve ter um login único e uma senha segura para acesso.

## 2.2 Requisitos de Relatórios

- **RR001 - Relatório de Medicamentos Mais Vendidos:**
  - **Descrição:** Gerar um relatório listando os medicamentos mais vendidos dentro de um período específico, mostrando a quantidade de vendas realizadas para cada medicamento.
- **RR002 - Relatório de Estoque Mínimo:**
  - **Descrição:** Criar um relatório que liste os medicamentos que estão com estoque abaixo de um nível mínimo pré-definido, indicando a necessidade de reabastecimento.
- **RR003 - Relatório de Vendas por Cliente:**
  - **Descrição:** Fornecer um relatório detalhado das vendas realizadas para cada cliente, incluindo o total de compras, medicamentos adquiridos e datas das transações.

## 3 Desenvolvimento

### 3.1 Entidades, Atributos e Relacionamentos

A seguir, apresentamos as principais entidades, seus atributos e relacionamentos para o Sistema de Gerenciamento de Remédios:

### 3.2 Entidade: Remédio

- id (identificador único)
- Nome

- Descrição

**Subgrupo da entidade remédio:**

- Entidade adicional: genérico
- Entidade adicional: de marca
- Entidade adicional: outras categorias
- Entidade adicional: Tarja

**Subgrupo da entidade remédio da entidade Tarja:**

- Atributo adicional: Cores da tarja

**Relacionamentos:**

- Um pedido pode conter vários remédios (Relacionamento 1:N)
- Um remédio pode estar presente em vários pedidos (Relacionamento N:1)

### **3.3 Entidade: Fornecedor**

- Nome da Empresa
- E-mail
- CNPJ
- Endereço
- Número
- Rua
- Bairro
- Cidade
- Estado



**Relacionamentos:**

- Um fornecedor pode fornecer vários pedidos (Relacionamento 1:N)
- Um pedido é feito para um único fornecedor (Relacionamento N:1)

**3.4 Entidade: Pedido**

- ID do Pedido (identificador único)
- Data do Pedido
- Quantidade de remédios solicitados
- Status do pedido
- Código de rastreamento

**Relacionamentos:**

- Um remédio pode estar associado a vários pedidos (Relacionamento N:1)
- Um pedido pode conter vários remédios (Relacionamento N:1)

**3.5 Entidade: Cliente**

- Nome
- CPF
- Telefone
- Endereço
- Número da casa
- Bairro
- Cidade
- Rua

**Relacionamentos:**

- Um cliente pode ter vários históricos de compras (Relacionamento 1:N)
- Cada compra é associada a um único cliente (Relacionamento 1:1)
- Cada cliente pode comprar vários remédios (Relacionamento N:M)
- Cada remédio pode ser comprado por vários clientes (Relacionamento N:M)

**3.6 Entidade: Funcionário**

- ID do Funcionário (identificador único)
- Nome
- Cargo
- Salário

**Relacionamentos:**

- Um funcionário pode ser responsável por vários pedidos (Relacionamento 1:N)
- Cada pedido é atribuído a um único funcionário (por exemplo, quem fez o pedido, quem processou, etc.).

**3.7 Entidade: Estoque**

- Data de validade
- Data de entrada no estoque
- Quantidade no estoque
- Unidade de medida
- Frascos
- Caixas
- Cartelas
- Unidade

## Relacionamentos:

- Um estoque pode conter vários tipos de remédios em diferentes quantidades (Relacionamento 1:N)
- Um remédio pode estar presente em vários estoques, cada um com sua própria quantidade (Relacionamento N:1)

## 4 Diagrama ER/EER

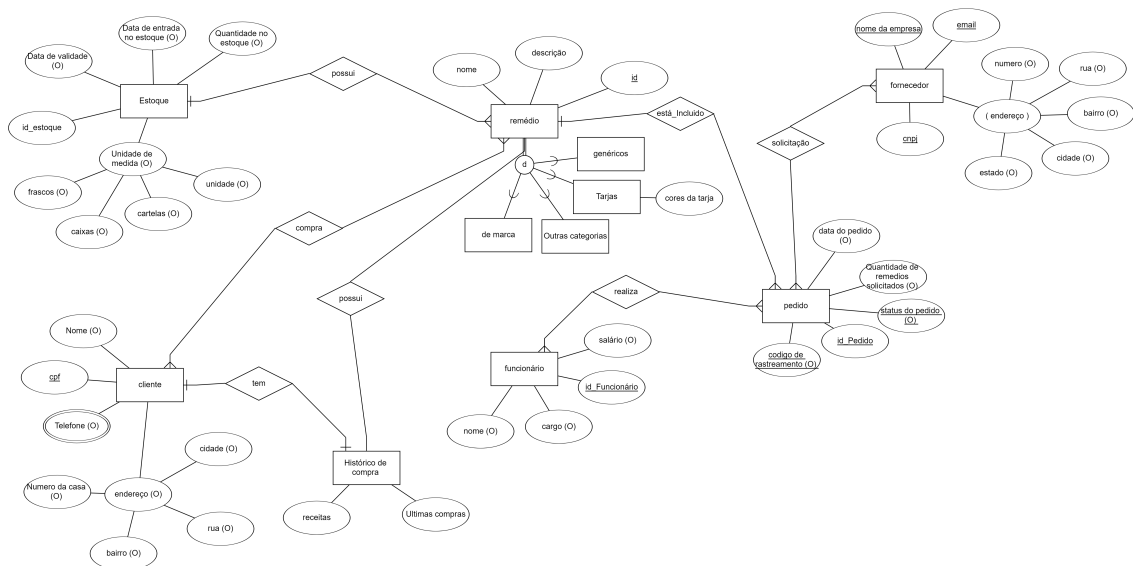


Figura 1: Diagrama ER/EER do Sistema de Gerenciamento de Remédios

## 5 Delegação de Atividades

Membro da Equipe	Atividade
Tailan	Desenvolvimento do modelo EER
Tailan	Análise final da etapa 1 do modelo EER
Kaio	Levantamento de requisitos do Sistema
Kaio	Documentação de Requisitos

Tabela 1: Delegação de Atividades

## 6 Conclusão

O projeto de banco de dados para o Sistema de Gerenciamento de remédios abrangeu a identificação das principais entidades e seus atributos, bem como a delegação de atividades entre os membros da equipe. Através dessa análise, foi possível estabelecer uma estrutura sólida para o desenvolvimento do sistema, visando facilitar o controle e a gestão de informações relacionadas a clientes, produtos, vendas e funcionários.

```
[12pt, a4paper]abntex2
graphicx longtable lscape
[alf]abntex2cite
geometry top=3cm, bottom=2cm, left=2cm, right=2cm
```

# Relatório de Projeto: Gerenciamento de Remédios

16 de agosto de 2024

## Resumo

Este relatório apresenta o desenvolvimento de um sistema para o gerenciamento de remédios, detalhando as principais tabelas do banco de dados e as operações CRUD implementadas na aplicação. O sistema utiliza o Flask para a construção da API e PostgreSQL como sistema de gerenciamento de banco de dados.

## Conteúdo

### 7 Introdução

O projeto tem como objetivo criar uma aplicação para o gerenciamento de remédios, com funcionalidades de CRUD para diferentes tabelas do banco de dados. A aplicação é construída utilizando Flask para a API e PostgreSQL para o banco de dados.

### 8 Estrutura do Banco de Dados

#### 8.1 Script para criação do banco de dados

```
CREATE SCHEMA farmacia;  
SET SCHEMA 'farmacia';  
  
-- Tabela Remédio  
CREATE TABLE remedio (  
    id_remedio SERIAL PRIMARY KEY,
```

```

        nome VARCHAR(255) NOT NULL,
        descricao VARCHAR(255) NOT NULL
    );

-- Tabela Genéricos, herda atributos da tabela Remédio
CREATE TABLE genericos (
    id_generico SERIAL PRIMARY KEY
) INHERITS (remedio);

-- Tabela Tarjas, herda atributos da tabela Remédio
CREATE TABLE tarjas (
    id_tarjas SERIAL PRIMARY KEY,
    cores_tarjas VARCHAR(255) NOT NULL
) INHERITS (remedio);

-- Tabela de Marcas, herda atributos da tabela Remédio
CREATE TABLE de_marcas (
    id_marcas SERIAL PRIMARY KEY
) INHERITS (remedio);

-- Tabela Pedidos
CREATE TABLE pedido (
    id_pedido SERIAL PRIMARY KEY,
    cod_rastr VARCHAR(50) UNIQUE,
    status_pedido VARCHAR(50) NOT NULL CHECK (status_pedido IN ('EM PRODUÇÃO', '
    data_pedido DATE,
    qtd_rem_solic INTEGER
);

-- Tabela Relacional entre Remédio e Pedido
CREATE TABLE esta_incluido (
    id_remedio INTEGER NOT NULL REFERENCES remedio (id_remedio),
    id_pedido INTEGER NOT NULL REFERENCES pedido (id_pedido),
    PRIMARY KEY (id_remedio, id_pedido)
);

-- Tabela Fornecedor
CREATE TABLE fornecedor (
    id_fornecedor SERIAL PRIMARY KEY,
    nome_empresa VARCHAR(255) UNIQUE NOT NULL,
    cnpj VARCHAR(14) UNIQUE NOT NULL,

```

```

        email VARCHAR(255),
        endereco_numero INTEGER,
        endereco_rua VARCHAR(255),
        endereco_bairro VARCHAR(255),
        endereco_cidade VARCHAR(255),
        endereco_estado CHAR(2)
    );

-- Tabela Relacional entre Pedido e Fornecedor
CREATE TABLE solicitacao (
    id_pedido INTEGER NOT NULL REFERENCES pedido (id_pedido),
    id_fornecedor INTEGER NOT NULL REFERENCES fornecedor (id_fornecedor),
    PRIMARY KEY (id_pedido, id_fornecedor)
);

-- Tabela Funcionário
CREATE TABLE funcionario (
    id_funcionario SERIAL PRIMARY KEY,
    salario NUMERIC(10, 2),
    cargo VARCHAR(255),
    nome VARCHAR(255)
);

-- Tabela Relacional entre Funcionário e Pedido
CREATE TABLE realiza (
    id_funcionario INTEGER NOT NULL REFERENCES funcionario (id_funcionario),
    id_pedido INTEGER NOT NULL REFERENCES pedido (id_pedido),
    PRIMARY KEY (id_funcionario, id_pedido)
);

-- Tabela Estoque
CREATE TABLE estoque (
    id_estoque SERIAL PRIMARY KEY,
    data_entrega_stq DATE,
    qtd_stq INTEGER,
    data_vali_stq DATE,
    medida_frascos INTEGER,
    medida_caixas INTEGER,
    medida_cartelas INTEGER,
    medida_unidade INTEGER
);

```

```

-- Tabela Relacional entre Estoque e Remédio
CREATE TABLE possui (
    id_estoque INTEGER NOT NULL REFERENCES estoque (id_estoque),
    id_remedio INTEGER NOT NULL REFERENCES remedio (id_remedio),
    PRIMARY KEY (id_estoque, id_remedio)
);

-- Tabela Cliente
CREATE TABLE cliente (
    id_cliente SERIAL PRIMARY KEY,
    nome_cliente VARCHAR(255),
    cpf CHAR(11) CHECK (char_length(cpf) = 11) NOT NULL,
    endereco_cliente_numero INTEGER,
    endereco_cliente_rua VARCHAR(255),
    endereco_cliente_bairro VARCHAR(255),
    endereco_cliente_cidade VARCHAR(255)
);

-- Tabela Telefone, relacionada ao atributo multivalorado de Cliente
CREATE TABLE telefone (
    id_telefone SERIAL PRIMARY KEY,
    id_cliente INTEGER NOT NULL REFERENCES cliente (id_cliente),
    numero_telefone VARCHAR(20) UNIQUE
);

-- Tabela Histórico de Compra
CREATE TABLE historico_compra (
    id_historico SERIAL PRIMARY KEY,
    receitas VARCHAR(255),
    ult_compra VARCHAR(255)
);

-- Tabela Relacional entre Histórico de Compra e Remédio
CREATE TABLE faz_parte (
    id_historico INTEGER NOT NULL REFERENCES historico_compra (id_historico),
    id_remedio INTEGER NOT NULL REFERENCES remedio (id_remedio),
    PRIMARY KEY (id_historico, id_remedio)
);

-- Tabela Relacional entre Histórico de Compra e Cliente

```



```

CREATE TABLE tem (
    id_historico INTEGER NOT NULL REFERENCES historico_compra (id_historico),
    id_cliente INTEGER NOT NULL REFERENCES cliente (id_cliente),
    PRIMARY KEY (id_historico, id_cliente)
);

```

## 8.2 Populando as tabela de maneira não aleatória

```

-- Populando a tabela Remédio
INSERT INTO remedio (nome, descricao) VALUES
('Paracetamol', 'Analgésico e antipirético'),
('Ibuprofeno', 'Anti-inflamatório e analgésico'),
('Amoxicilina', 'Antibiótico de amplo espectro'),
('Dipirona', 'Analgésico e antipirético'),
('Aspirina', 'Anti-inflamatório e antipirético'),
('Losartana', 'Antagonista dos receptores da angiotensina II'),
('Clonazepam', 'Benzodiazepínico usado como anticonvulsivante'),
('Omeprazol', 'Inibidor de bomba de prótons, usado para úlceras gástricas'),
('Simvastatina', 'Inibidor de HMG-CoA redutase, usado para baixar colesterol'),
('Metformina', 'Antidiabético oral, usado no tratamento de diabetes tipo 2');

-- Populando a tabela Genéricos
INSERT INTO genericos (nome, descricao) VALUES
('Paracetamol Genérico', 'Analgésico e antipirético'),
('Ibuprofeno Genérico', 'Anti-inflamatório e analgésico'),
('Amoxicilina Genérica', 'Antibiótico de amplo espectro'),
('Dipirona Genérica', 'Analgésico e antipirético'),
('Aspirina Genérica', 'Anti-inflamatório e antipirético'),
('Losartana Genérica', 'Antagonista dos receptores da angiotensina II'),
('Clonazepam Genérico', 'Benzodiazepínico usado como anticonvulsivante'),
('Omeprazol Genérico', 'Inibidor de bomba de prótons, usado para úlceras gástricas'),
('Simvastatina Genérica', 'Inibidor de HMG-CoA redutase, usado para baixar colesterol'),
('Metformina Genérica', 'Antidiabético oral, usado no tratamento de diabetes tipo 2');

-- Populando a tabela Tarjas
INSERT INTO tarjas (nome, descricao, cores_tarjas) VALUES
('Paracetamol', 'Analgésico e antipirético', 'Vermelha'),
('Ibuprofeno', 'Anti-inflamatório e analgésico', 'Vermelha'),
('Amoxicilina', 'Antibiótico de amplo espectro', 'Amarela'),

```

```

('Dipirona', 'Analgésico e antipirético', 'Vermelha'),
('Aspirina', 'Anti-inflamatório e antipirético', 'Amarela'),
('Losartana', 'Antagonista dos receptores da angiotensina II', 'Vermelha'),
('Clonazepam', 'Benzodiazepínico usado como anticonvulsivante', 'Preta'),
('Omeprazol', 'Inibidor de bomba de prótons, usado para úlceras gástricas', 'Vermelha'),
('Simvastatina', 'Inibidor de HMG-CoA redutase, usado para baixar colesterol', 'Vermelha'),
('Metformina', 'Antidiabético oral, usado no tratamento de diabetes tipo 2', 'Vermelha'),

-- Populando a tabela de Marcas
INSERT INTO de_marcas (nome, descricao) VALUES
('Tylenol', 'Analgésico e antipirético'),
('Advil', 'Anti-inflamatório e analgésico'),
('Amoxil', 'Antibiótico de amplo espectro'),
('Novalgina', 'Analgésico e antipirético'),
('Aspirina Bayer', 'Anti-inflamatório e antipirético'),
('Cozaar', 'Antagonista dos receptores da angiotensina II'),
('Rivotril', 'Benzodiazepínico usado como anticonvulsivante'),
('Losec', 'Inibidor de bomba de prótons, usado para úlceras gástricas'),
('Zocor', 'Inibidor de HMG-CoA redutase, usado para baixar colesterol'),
('Glyciphage', 'Antidiabético oral, usado no tratamento de diabetes tipo 2');

-- Populando a tabela Pedidos
INSERT INTO pedido (cod_rastr, status_pedido, data_pedido, qtd_rem_solic) VALUES
('RAST001', 'EM PRODUÇÃO', '2024-08-01', 100),
('RAST002', 'A CAMINHO', '2024-08-02', 50),
('RAST003', 'ENTREGUE', '2024-08-03', 150),
('RAST004', 'A CAMINHO', '2024-08-04', 200),
('RAST005', 'EM PRODUÇÃO', '2024-08-05', 80),
('RAST006', 'ENTREGUE', '2024-08-06', 60),
('RAST007', 'A CAMINHO', '2024-08-07', 90),
('RAST008', 'ENTREGUE', '2024-08-08', 70),
('RAST009', 'EM PRODUÇÃO', '2024-08-09', 110),
('RAST010', 'A CAMINHO', '2024-08-10', 120);

-- Populando a tabela Esta Incluído
INSERT INTO esta_incluido (id_remedio, id_pedido) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),

```

```
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10);
```

```
-- Populando a tabela Fornecedor
```

```
INSERT INTO fornecedor (nome_empresa, cnpj, email, endereco_numero, endereco_rua)
VALUES
('Farmácia São João', '12345678000101', 'contato@saovao.com', 100, 'Rua Principal'),
('Droga Raia', '98765432000109', 'contato@droga.com', 200, 'Avenida Central', 'Centro'),
('Panvel', '11223344000155', 'contato@panvel.com', 300, 'Rua das Flores', 'Jardim'),
('Pacheco', '99887766000102', 'contato@pacheco.com', 400, 'Avenida Paulista', 'Centro'),
('Onofre', '55667788000111', 'contato@onofre.com', 500, 'Rua Augusta', 'Centro'),
('ExtraFarma', '44332211000108', 'contato@extra.com', 600, 'Avenida Brasil', 'Vila'),
('Pague Menos', '66778899000144', 'contato@pague.com', 700, 'Rua da Praia', 'Centro'),
('Ultrafarma', '33556677000122', 'contato@ultra.com', 800, 'Avenida Atlântica', 'Centro'),
('Drogasil', '77441122000133', 'contato@drogasil.com', 900, 'Rua XV de Novembro', 'Centro'),
('Venâncio', '22113344000166', 'contato@venancio.com', 1000, 'Rua da Alfândega', 'Centro');
```

```
-- Populando a tabela Solicitação
```

```
INSERT INTO solicitacao (id_pedido, id_fornecedor) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10);
```

```
-- Populando a tabela Funcionário
```

```
INSERT INTO funcionario (salario, cargo, nome) VALUES
(3500.00, 'Atendente', 'João Silva'),
(4500.00, 'Farmacêutico', 'Maria Oliveira'),
(5000.00, 'Gerente', 'Carlos Souza'),
(3000.00, 'Caixa', 'Ana Lima'),
(3800.00, 'Atendente', 'Pedro Mendes'),
(4200.00, 'Farmacêutico', 'Luísa Fernandes'),
(5500.00, 'Gerente', 'Clara Costa');
```

```
(2900.00, 'Caixa', 'Bruno Pereira'),
(3700.00, 'Atendente', 'Fernanda Santos'),
(4400.00, 'Farmacêutico', 'Lucas Almeida');
```

```
-- Populando a tabela Realiza
```

```
INSERT INTO realiza (id_funcionario, id_pedido) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10);
```

```
-- Populando a tabela Estoque
```

```
INSERT INTO estoque (data_entrega_stq, qtd_stq, data_vali_stq, medida_frascos, m
('2024-07-01', 100, '2025-07-01', 50, 30, 10, 10),
('2024-07-02', 200, '2025-07-02', 60, 40, 20, 20),
('2024-07-03', 300, '2025-07-03', 70, 50, 30, 30),
('2024-07-04', 400, '2025-07-04', 80, 60, 40, 40),
('2024-07-05', 500, '2025-07-05', 90, 70, 50, 50),
('2024-07-06', 600, '2025-07-06', 100, 80, 60, 60),
('2024-07-07', 700, '2025-07-07', 110, 90, 70, 70),
('2024-07-08', 800, '2025-07-08', 120, 100, 80, 80),
('2024-07-09', 900, '2025-07-09', 130, 110, 90, 90),
('2024-07-10', 1000, '2025-07-10', 140, 120, 100, 100);
```

```
-- Populando a tabela Possui
```

```
INSERT INTO possui (id_estoque, id_remedio) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
```

```
(10, 10);
```

```
-- Populando a tabela Cliente
```

```
INSERT INTO cliente (nome_cliente, cpf, endereco_cliente_numero, endereco_cliente_cidade, endereco_cliente_estado) VALUES  
( 'Carlos Almeida', '12345678901', 100, 'Rua A', 'Centro', 'São Paulo'),  
( 'Mariana Santos', '98765432102', 200, 'Rua B', 'Jardim', 'Rio de Janeiro'),  
( 'Roberto Costa', '11122233344', 300, 'Rua C', 'Vila Mariana', 'Salvador'),  
( 'Ana Paula', '55566677788', 400, 'Rua D', 'Copacabana', 'Belo Horizonte'),  
( 'João Pedro', '99988877766', 500, 'Rua E', 'Centro', 'Porto Alegre'),  
( 'Fernanda Lima', '33344455522', 600, 'Rua F', 'Centro', 'Curitiba'),  
( 'Ricardo Oliveira', '77788899900', 700, 'Rua G', 'Cidade Baixa', 'Recife'),  
( 'Patrícia Mendes', '22233344411', 800, 'Rua H', 'Jardim', 'Fortaleza'),  
( 'Lucas Pereira', '66655544433', 900, 'Rua I', 'Centro', 'São Luís'),  
( 'Juliana Fernandes', '88877766655', 1000, 'Rua J', 'Centro', 'Manaus');
```

```
-- Populando a tabela Telefone
```

```
INSERT INTO telefone (id_cliente, numero_telefone) VALUES  
(1, '(11) 99999-0001'),  
(2, '(21) 99999-0002'),  
(3, '(71) 99999-0003'),  
(4, '(31) 99999-0004'),  
(5, '(51) 99999-0005'),  
(6, '(41) 99999-0006'),  
(7, '(81) 99999-0007'),  
(8, '(85) 99999-0008'),  
(9, '(98) 99999-0009'),  
(10, '(92) 99999-0010');
```

```
-- Populando a tabela Histórico de Compra
```

```
INSERT INTO historico_compra (receitas, ult_compra) VALUES  
( 'Receita para antibiótico', '2024-06-15'),  
( 'Receita para anti-inflamatório', '2024-07-01'),  
( 'Receita para analgésico', '2024-07-05'),  
( 'Receita para antiácido', '2024-07-10'),  
( 'Receita para antidiabético', '2024-07-15'),  
( 'Receita para ansiolítico', '2024-07-20'),  
( 'Receita para anti-hipertensivo', '2024-07-25'),  
( 'Receita para antibiótico', '2024-07-30'),  
( 'Receita para antitérmico', '2024-08-05'),  
( 'Receita para hipoglicemiante', '2024-08-10');
```

```

-- Populando a tabela Faz Parte
INSERT INTO faz_parte (id_historico, id_remedio) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10);

-- Populando a tabela Tem
INSERT INTO tem (id_historico, id_cliente) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10);

```

### **8.3 Ver todas tabelas com os dados informados;**

```

-- Visualizar o conteúdo da tabela Remédio
SELECT * FROM farmacia.remedio;

-- Visualizar o conteúdo da tabela Genéricos
SELECT * FROM farmacia.genericos;

-- Visualizar o conteúdo da tabela Tarjas
SELECT * FROM farmacia.tarjas;

-- Visualizar o conteúdo da tabela Marcas
SELECT * FROM farmacia.de_marcas;

```

```
-- Visualizar o conteúdo da tabela Pedidos
SELECT * FROM farmacia.pedido;

-- Visualizar o conteúdo da tabela Esta Incluído
SELECT * FROM farmacia.esta_incluido;

-- Visualizar o conteúdo da tabela Fornecedor
SELECT * FROM farmacia.fornecedor;

-- Visualizar o conteúdo da tabela Solicitação
SELECT * FROM farmacia.solicitacao;

-- Visualizar o conteúdo da tabela Funcionário
SELECT * FROM farmacia.funcionario;

-- Visualizar o conteúdo da tabela Realiza
SELECT * FROM farmacia.realiza;

-- Visualizar o conteúdo da tabela Estoque
SELECT * FROM farmacia.estoque;

-- Visualizar o conteúdo da tabela Possui
SELECT * FROM farmacia.possui;

-- Visualizar o conteúdo da tabela Cliente
SELECT * FROM farmacia.cliente;

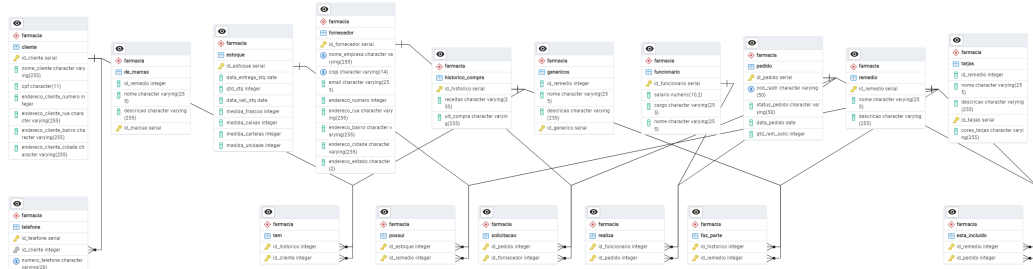
-- Visualizar o conteúdo da tabela Telefone
SELECT * FROM farmacia.telefone;

-- Visualizar o conteúdo da tabela Histórico de Compra
SELECT * FROM farmacia.historico_compra;

-- Visualizar o conteúdo da tabela Faz Parte
SELECT * FROM farmacia.faz_parte;

-- Visualizar o conteúdo da tabela Tem
SELECT * FROM farmacia.tem;
```

#### 8.4 Logo abaixo modelo relacional do projeto de gerenciamento de remédios



**9 O banco de dados é estruturado com as seguintes tabelas principais:**

## 9.1 Tabela remedio

- **Descrição:** Armazena informações sobre os remédios disponíveis.
- **Chave Primária:** `id_remedio`
- **Atributos:**
  - `id_remedio`: Identificador único para cada remédio.
  - `nome`: Nome do remédio.
  - `descricao`: Descrição do remédio.

## 9.2 Tabela genericos

- **Descrição:** Armazena informações sobre remédios genéricos.
- **Chave Primária:** `id_generico`
- **Herança:** Herda todos os atributos de `remedio`.
- **Atributos:**
  - `id_generico`: Identificador único para cada remédio genérico.



### 9.3 Tabela `tarjas`

- **Descrição:** Armazena informações sobre remédios com tarjas.
- **Chave Primária:** `id_tarjas`
- **Herança:** Herda todos os atributos de `remedio`.
- **Atributos:**
  - `id_tarjas`: Identificador único para cada remédio com tarja.
  - `cores_tarjas`: Cores das tarjas do remédio.

### 9.4 Tabela `de_marcas`

- **Descrição:** Armazena informações sobre remédios de marcas específicas.
- **Chave Primária:** `id_marcas`
- **Herança:** Herda todos os atributos de `remedio`.
- **Atributos:**
  - `id_marcas`: Identificador único para cada remédio de marca.

### 9.5 Tabela `pedido`

- **Descrição:** Armazena informações sobre pedidos realizados pela farmácia.
- **Chave Primária:** `id_pedido`
- **Chave Candidata:** `cod_rastr`
- **Atributos:**
  - `id_pedido`: Identificador único para cada pedido.
  - `cod_rastr`: Código de rastreamento do pedido.
  - `status_pedido`: Status do pedido, que pode ser EM PRODUÇÃO, A CAMINHO, ou ENTREGUE.
  - `data_pedido`: Data do pedido.
  - `qtd_rem_solic`: Quantidade de remédios solicitada.

## 9.6 Tabela esta\_incluido

- **Descrição:** Relaciona os remédios incluídos em um pedido específico.
- **Chave Primária:** id\_remedio, id\_pedido
- **Chaves Estrangeiras:**
  - id\_remedio: Referencia a tabela remedio.
  - id\_pedido: Referencia a tabela pedido.

## 9.7 Tabela fornecedor

- **Descrição:** Armazena informações sobre fornecedores.
- **Chave Primária:** id\_fornecedor
- **Chaves Candidatas:** nome\_empresa, cnpj
- **Atributos:**
  - id\_fornecedor: Identificador único para cada fornecedor.
  - nome\_empresa: Nome da empresa fornecedora.
  - cnpj: CNPJ da empresa fornecedora.
  - email: Email da empresa fornecedora.
  - endereco\_numero: Número do endereço da empresa.
  - endereco\_rua: Rua do endereço da empresa.
  - endereco\_bairro: Bairro do endereço da empresa.
  - endereco\_cidade: Cidade do endereço da empresa.
  - endereco\_estado: Estado do endereço da empresa (2 caracteres).

## 9.8 Tabela solicitacao

- **Descrição:** Relaciona os pedidos com os fornecedores.
- **Chave Primária:** id\_pedido, id\_fornecedor
- **Chaves Estrangeiras:**
  - id\_pedido: Referencia a tabela pedido.
  - id\_fornecedor: Referencia a tabela fornecedor.

## 9.9 Tabela funcionario

- **Descrição:** Armazena informações sobre os funcionários.
- **Chave Primária:** id.funcionario
- **Atributos:**
  - id.funcionario: Identificador único para cada funcionário.
  - salario: Salário do funcionário.
  - cargo: Cargo do funcionário.
  - nome: Nome do funcionário.

## 9.10 Tabela realiza

- **Descrição:** Relaciona funcionários com os pedidos que realizam.
- **Chave Primária:** id.funcionario, id\_pedido
- **Chaves Estrangeiras:**
  - id.funcionario: Referencia a tabela funcionario.
  - id\_pedido: Referencia a tabela pedido.

## 9.11 Tabela estoque

- **Descrição:** Armazena informações sobre o estoque de remédios.
- **Chave Primária:** id.estoque
- **Atributos:**
  - id.estoque: Identificador único para cada registro de estoque.
  - data\_entrega\_stq: Data de entrega ao estoque.
  - qtd\_stq: Quantidade no estoque.
  - data\_vali\_stq: Data de validade do estoque.
  - medida\_frascos: Quantidade em frascos.
  - medida\_caixas: Quantidade em caixas.
  - medida\_cartelas: Quantidade em cartelas.
  - medida\_unidade: Quantidade em unidades.

### 9.12 Tabela possui

- **Descrição:** Relaciona remédios com o estoque.
- **Chave Primária:** `id_estoque`, `id_remedio`
- **Chaves Estrangeiras:**
  - `id_estoque`: Referencia a tabela `estoque`.
  - `id_remedio`: Referencia a tabela `remedio`.

## 10 Descrição das Tabelas

### 10.1 Tabela cliente

- **Descrição:** Armazena informações sobre os clientes, incluindo dados pessoais e endereço.
- **Chave Primária:** `id_cliente`
- **Chaves Candidatas:**
  - `cpf`
- **Atributos:**
  - `id_cliente`: Identificador único do cliente.
  - `nome_cliente`: Nome do cliente.
  - `cpf`: CPF do cliente (único).
  - `endereco_cliente_numero`: Número do endereço do cliente.
  - `endereco_cliente_rua`: Rua do endereço do cliente.
  - `endereco_cliente_bairro`: Bairro do endereço do cliente.
  - `endereco_cliente_cidade`: Cidade do endereço do cliente.

### 10.2 Tabela telefone

- **Descrição:** Armazena números de telefone associados aos clientes. Esta tabela representa um atributo multivalorado da tabela `cliente`.
- **Chave Primária:** `id_telefone`
- **Chaves Estrangeiras:**

- `id_cliente`: Referencia a tabela `cliente`.

- **Atributos:**

- `id_telefone`: Identificador único do telefone.
- `id_cliente`: Identificador do cliente a quem o telefone pertence.
- `numero_telefone`: Número do telefone (único).

### 10.3 Tabela `historico_compra`

- **Descrição:** Armazena o histórico de compras dos clientes, incluindo informações sobre receitas e última compra.
- **Chave Primária:** `id_historico`
- **Atributos:**
  - `id_historico`: Identificador único do histórico.
  - `receitas`: Descrição das receitas associadas ao histórico de compra.
  - `ult_compra`: Descrição da última compra realizada.

### 10.4 Tabela `faz_parte`

- **Descrição:** Relaciona o histórico de compra com os remédios comprados.
- **Chave Primária:** `id_historico`, `id_remedio`
- **Chaves Estrangeiras:**
  - `id_historico`: Referencia a tabela `historico_compra`.
  - `id_remedio`: Referencia a tabela `remedio`.

### 10.5 Tabela `tem`

- **Descrição:** Relaciona o histórico de compra com os clientes que realizaram a compra.
- **Chave Primária:** `id_historico`, `id_cliente`
- **Chaves Estrangeiras:**
  - `id_historico`: Referencia a tabela `historico_compra`.
  - `id_cliente`: Referencia a tabela `cliente`.

## 11 API e Funcionalidades

A API foi desenvolvida utilizando Flask e inclui endpoints para operações CRUD para todas as tabelas mencionadas. A seguir, estão listadas algumas das rotas implementadas:

### 11.1 CRUD para Tabela Remédio

- **GET** /remedio: Lista todos os remédios.
- **POST** /remedio: Adiciona um novo remédio.
- **PUT** /remedio/id: Atualiza um remédio existente.
- **DELETE** /remedio/id: Exclui um remédio.

### 11.2 CRUD para Tabela Fornecedor

- **GET** /fornecedor: Lista todos os fornecedores.
- **POST** /fornecedor: Adiciona um novo fornecedor.
- **PUT** /fornecedor/id: Atualiza um fornecedor existente.
- **DELETE** /fornecedor/id: Exclui um fornecedor.

## 12 Dado a criação do banco de dados e de ter populado suas tabelas de maneiras não aleatória, aqui está o resultado de maneira individual

article graphicx

## 13 Exemplos de Implementação

### 13.1 Atributos Multivalorados

No banco de dados, o atributo multivalorado foi implementado através da tabela `telefone`. Esta tabela permite que um cliente tenha vários números de telefone associados a ele, cada um armazenado em um registro distinto.

Por exemplo:

```
CREATE TABLE telefone (
    id_telefone SERIAL,
    id_cliente INTEGER NOT NULL,
    numero_telefone VARCHAR(20) UNIQUE,
    PRIMARY KEY (id_telefone),
    FOREIGN KEY (id_cliente) REFERENCES cliente (id_cliente)
);
```

Neste exemplo, a tabela **telefone** possui uma chave estrangeira **id\_cliente** que referencia a tabela **cliente**. Isso permite que múltiplos números de telefone sejam associados a um único cliente, implementando assim o conceito de atributo multivalorado.

## 13.2 Hierarquia de Herança

A hierarquia de herança foi implementada utilizando o conceito de herança de tabelas. No SQL, isso foi realizado utilizando a palavra-chave **INHERITS**.

Por exemplo:

```
CREATE TABLE genericos (
    id_generico SERIAL,
    PRIMARY KEY (id_generico)
) INHERITS (remedio);
```

Neste exemplo, a tabela **genericos** herda todos os atributos da tabela **remedio**. Isso significa que qualquer registro na tabela **genericos** terá os mesmos campos que um registro na tabela **remedio**, além dos campos específicos definidos em **genericos**.

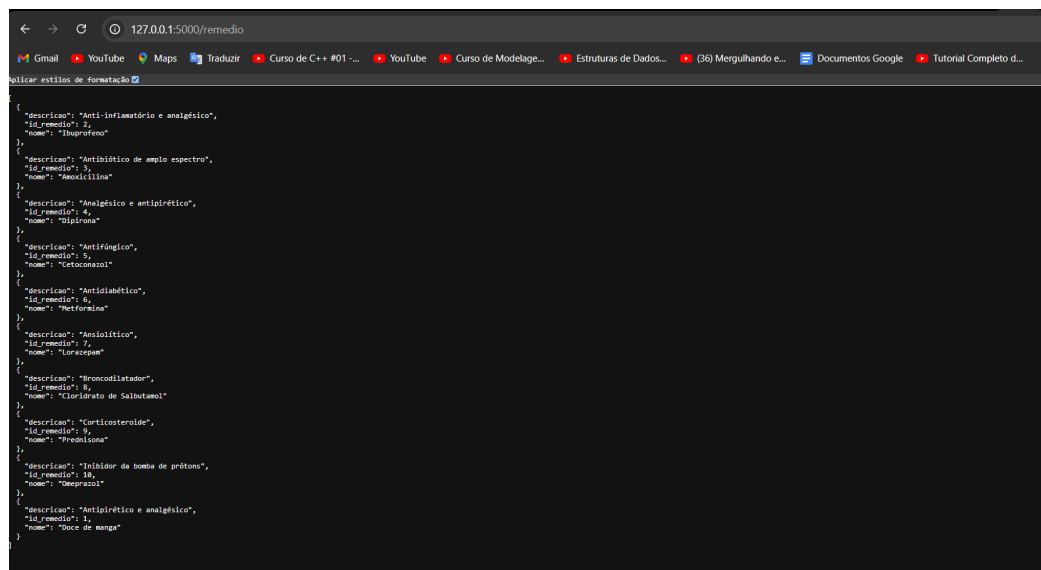
## 13.3 Conclusão

Os exemplos fornecidos demonstram como atributos multivalorados e a hierarquia de herança foram implementados no modelo relacional do banco de dados. Através do uso de tabelas separadas para atributos multivalorados e da herança de tabelas para modelar hierarquias, o banco de dados mantém a integridade e flexibilidade necessárias para gerenciar as informações de maneira eficiente.

## 14 Como Funciona um CRUD

CRUD é um acrônimo para as operações básicas de gerenciamento de dados: Create (Criar), Read (Ler), Update (Atualizar) e Delete (Excluir). Aqui estão exemplos de como cada operação é realizada utilizando Flask e PostgreSQL.

### 14.1 Imagem original sem nenhuma implementação (Tabela remédios)



### 14.2 Create (Criar)

Para adicionar um novo registro à tabela, você usa a operação POST. A seguir está um exemplo de código para adicionar um novo remédio:

Usando o POSTMAN para gerenciar esses dados;  
Com o campo "POST" `http://localhost:5000/remedio/11`  
no body;

```
{
  "descricao": "Antipirético e analgésico",
  "id_remedio": 11,
  "nome": "Manga doce"
}
```



HTTP **http://localhost:5000/remedio/11** Save Share

**POST** **http://localhost:5000/remedio/11** Send

Params Auth Headers (8) **Body** Scripts Tests Settings Cookies Beautify

raw JSON

```

1 {
2   .... "descricao": "Antipirético e analgésico",
3   .... "id_remédio": 11,
4   .... "nome": "Manga doce"
5 }

```

Gerenciamento Remedios/postgres@PostgreSQL 16

Query Query History

```
1 select * from farmacia.remedio
```

Data Output Messages Notifications

	id_remédio [PK] integer	nome character varying (255)	descricao character varying (255)
1	2	Ibuprofeno	Anti-inflamatório e analgésico
2	3	Amoxicilina	Antibiótico de amplo espectro
3	4	Dipirona	Analgésico e antipirético
4	5	Cetoconazol	Antifúngico
5	6	Metformina	Antidiabético
6	7	Lorazepam	Ansiolítico
7	8	Cloridrato de Salbutamol	Broncodilatador
8	9	Prednisona	Corticosteroide
9	10	Omeprazol	Inibidor da bomba de prótons
10	1	Paracetamol	Antipirético e analgésico
11	11	Manga doce	Antipirético e analgésico

Total rows: 11 of 11 Query complete 00:00:00.084 Ln 1, Col 31

```

@app.route('/remedio', methods=['POST'])
def adicionar_remedio():
    conn = get_db_connection()
    if conn is None:
        return jsonify({"error": "Não foi possível conectar ao banco de dados"})

    data = request.json
    nome = data['nome']

```

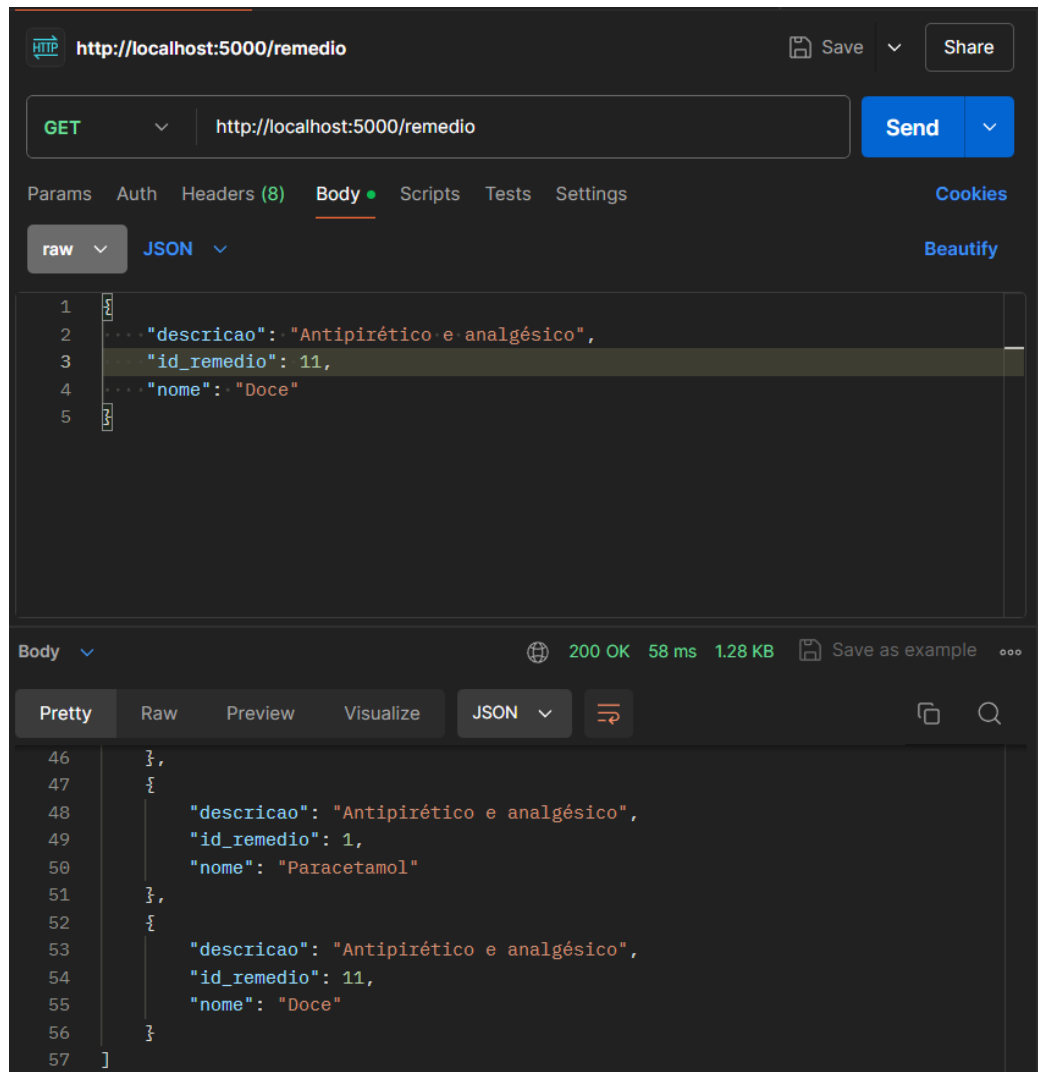
```

descricao = data['descricao']

try:
    with conn.cursor() as cur:
        cur.execute(
            "INSERT INTO farmacia.remedio (nome, descricao) VALUES (%s, %s) "
            (nome, descricao)
        )
        id_remedio = cur.fetchone()[0]
        conn.commit()
        return jsonify({"id_remedio": id_remedio}), 201
except Exception as e:
    return jsonify({"error": str(e)}), 500
finally:
    conn.close()

```

## 14.3 Após usar o CREATE



## 14.4 Read (Ler)

Para obter dados existentes, você usa a operação GET. Aqui está um exemplo de código para listar todos os remédios:

Usando o POSTMAN para gerenciar esses dados;  
Com o campo "GET" `http://localhost:5000/remedio/11`

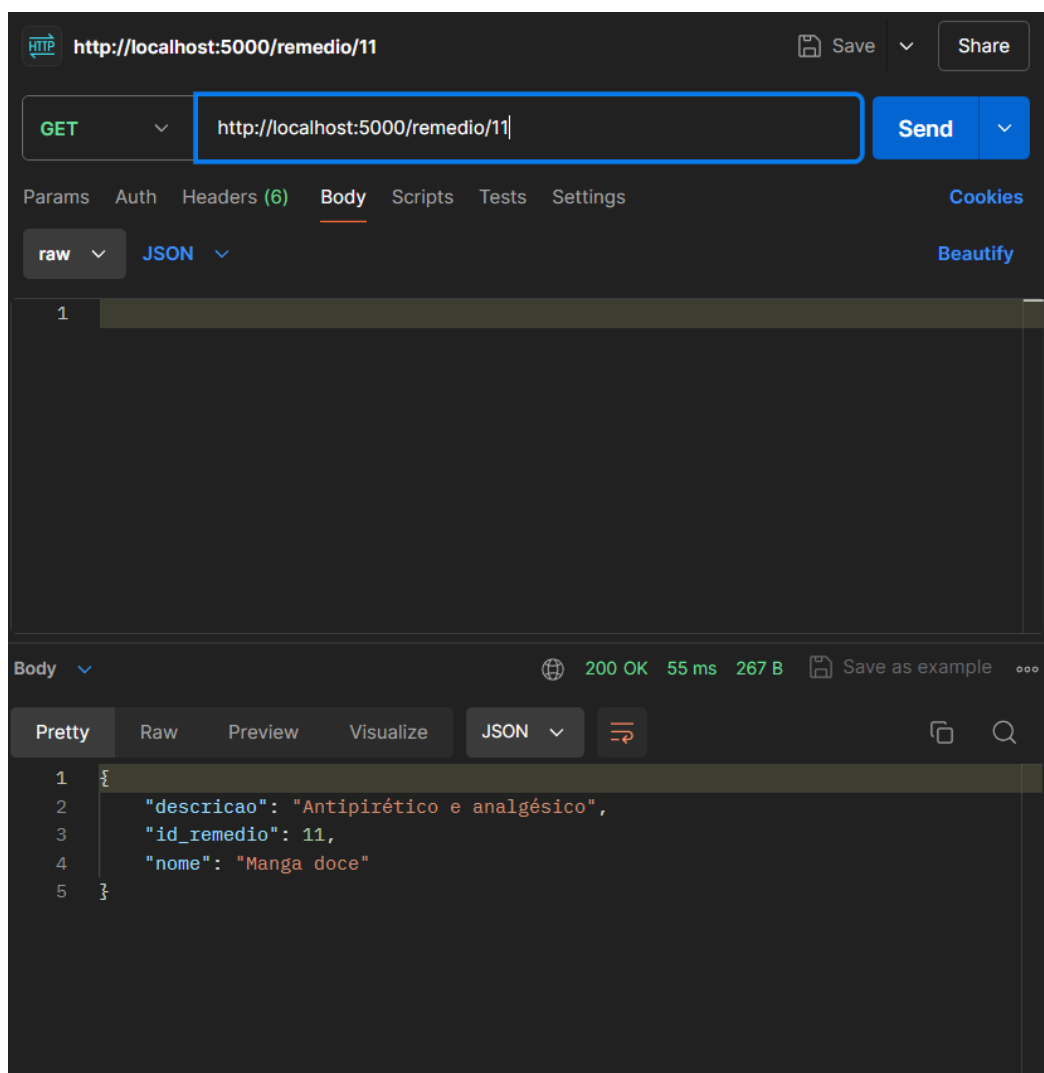
```
@app.route('/remedio', methods=['GET'])
def listar_remedios():
```

```

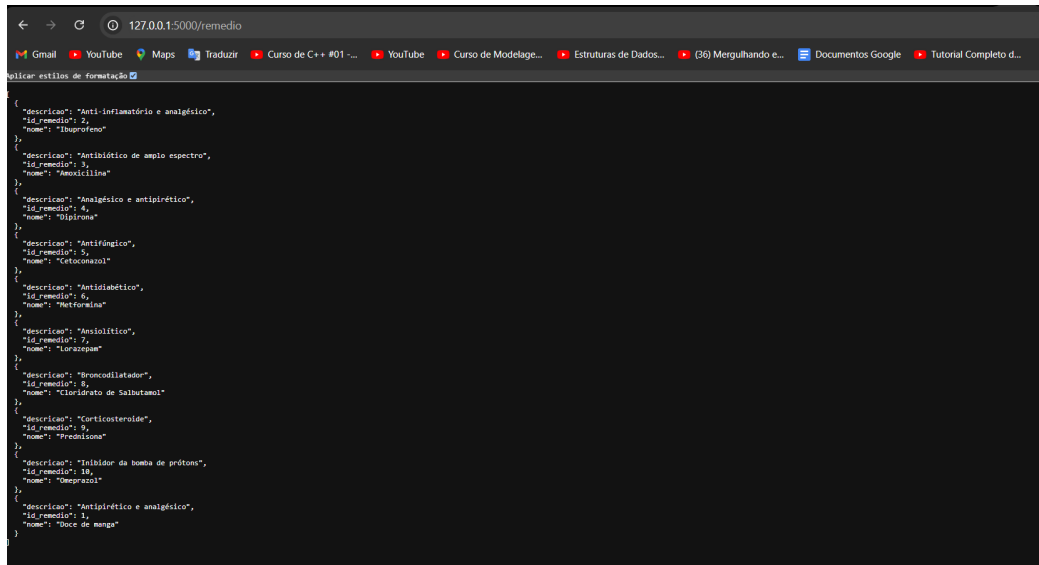
conn = get_db_connection()
if conn is None:
    return jsonify({"error": "Não foi possível conectar ao banco de dados"})

try:
    with conn.cursor() as cur:
        cur.execute("SELECT * FROM farmacia.remedio;")
        dados = cur.fetchall()
        colunas = [desc[0] for desc in cur.description]
        resultados = [dict(zip(colunas, linha)) for linha in dados]
        return jsonify(resultados)
except Exception as e:
    return jsonify({"error": str(e)}), 500
finally:
    conn.close()

```

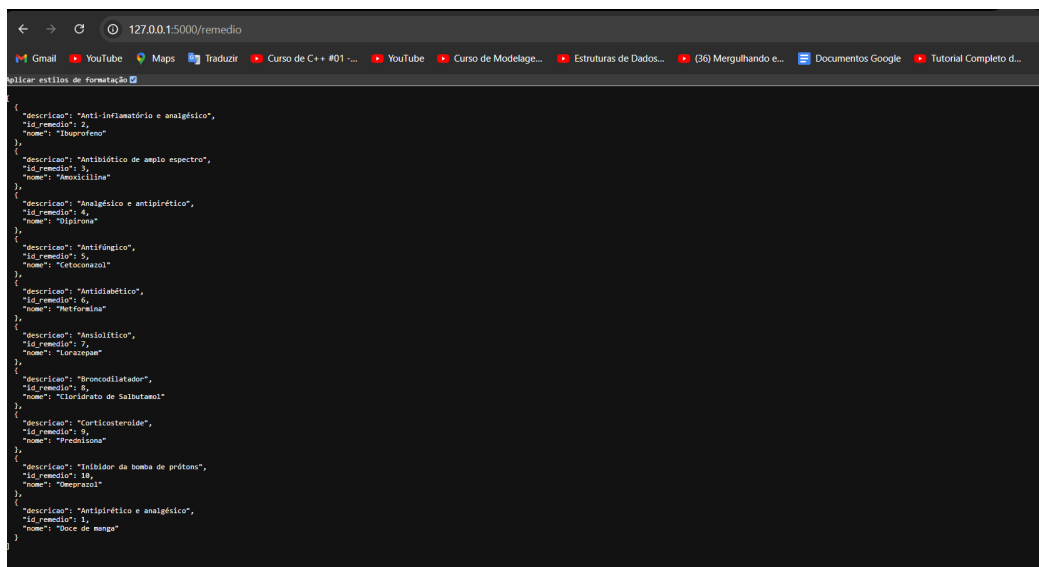


## 14.5 Resultando após usar o READ



```
{
  "descricao": "Anti-inflamatório e analgésico",
  "id_remedio": 2,
  "nome": "Ibuprofeno"
},
{
  "descricao": "Antibiótico de amplo espectro",
  "id_remedio": 3,
  "nome": "Amoxicilina"
},
{
  "descricao": "Analgésico e antipirético",
  "id_remedio": 4,
  "nome": "Dipirona"
},
{
  "descricao": "Antifúngico",
  "id_remedio": 5,
  "nome": "Cetoconazol"
},
{
  "descricao": "Antidiabético",
  "id_remedio": 6,
  "nome": "Metformina"
},
{
  "descricao": "Ansiolítico",
  "id_remedio": 7,
  "nome": "Lorazepam"
},
{
  "descricao": "Broncodilatador",
  "id_remedio": 8,
  "nome": "Cloridrato de Salbutamol"
},
{
  "descricao": "Corticosteroide",
  "id_remedio": 9,
  "nome": "Prednisona"
},
{
  "descricao": "Inibidor da bomba de prótons",
  "id_remedio": 10,
  "nome": "Omeprazol"
},
{
  "descricao": "Antipirético e analgésico",
  "id_remedio": 1,
  "nome": "Doce de manga"
}
```

## 14.6 Antes de utilizar o UPDATE



```
{
  "descricao": "Anti-inflamatório e analgésico",
  "id_remedio": 2,
  "nome": "Ibuprofeno"
},
{
  "descricao": "Antibiótico de amplo espectro",
  "id_remedio": 3,
  "nome": "Amoxicilina"
},
{
  "descricao": "Analgésico e antipirético",
  "id_remedio": 4,
  "nome": "Dipirona"
},
{
  "descricao": "Antifúngico",
  "id_remedio": 5,
  "nome": "Cetoconazol"
},
{
  "descricao": "Antidiabético",
  "id_remedio": 6,
  "nome": "Metformina"
},
{
  "descricao": "Ansiolítico",
  "id_remedio": 7,
  "nome": "Lorazepam"
},
{
  "descricao": "Broncodilatador",
  "id_remedio": 8,
  "nome": "Cloridrato de Salbutamol"
},
{
  "descricao": "Corticosteroide",
  "id_remedio": 9,
  "nome": "Prednisona"
},
{
  "descricao": "Inibidor da bomba de prótons",
  "id_remedio": 10,
  "nome": "Omeprazol"
},
{
  "descricao": "Antipirético e analgésico",
  "id_remedio": 1,
  "nome": "Doce de manga"
}
```

## 14.7 Update (Atualizar)

Para atualizar um registro existente, você usa a operação PUT. Veja um exemplo para atualizar um remédio:

Usando o POSTMAN para gerenciar esses dados;

Com o campo "PUT" `http://localhost:5000/remedio/11`  
no body;

```
{
  "descricao": "Antipirético e analgésico",
  "id_remedio": 11,
  "nome": "Doce de manga"
}
```

Gerenciamento Remedios/postgres@PostgreSQL 16

Query Query History

```
1 select * from farmacia.remedio
```

Data Output Messages Notifications

	id_remedio [PK] integer	nome character varying (255)	descricao character varying (255)
1	2	Ibuprofeno	Anti-inflamatório e analgésico
2	3	Amoxicilina	Antibiótico de amplo espectro
3	4	Dipirona	Analgésico e antipirético
4	5	Cetoconazol	Antifúngico
5	6	Metformina	Antidiabético
6	7	Lorazepam	Ansiolítico
7	8	Cloridrato de Salbutamol	Broncodilatador
8	9	Prednisona	Corticosteroide
9	10	Omeprazol	Inibidor da bomba de prótons
10	1	Paracetamol	Antipirético e analgésico
11	11	Manga doce	Antipirético e analgésico

Total rows: 11 of 11 Query complete 00:00:00.084 Ln 1, Col 31

HTTP <http://localhost:5000/remedio/11> Save Share

**PUT** <http://localhost:5000/remedio/11> Send

Params Auth Headers (8) **Body** Scripts Tests Settings Cookies

raw JSON Beautify

```
1 {
2   ... "descricao": "Antipirético e analgésico",
3   ... "id_remedio": 11,
4   ... "nome": "Doce de manga"
5 }
```

Body 200 OK 55 ms 217 B Save as example

Pretty Raw Preview Visualize JSON Copy Search

```
1 {
2   "message": "Operação bem-sucedida!"
3 }
```



Query Query History			
1 select * from farmacia.remedio			
Data Output Messages Notifications			
	id_remedio [PK] integer	nome character varying (255)	descricao character varying (255)
1	2	Ibuprofeno	Anti-inflamatório e analgésico
2	3	Amoxicilina	Antibiótico de amplo espectro
3	4	Dipirona	Analgésico e antipirético
4	5	Cetoconazol	Antifúngico
5	6	Metformina	Antidiabético
6	7	Lorazepam	Ansiolítico
7	8	Cloridrato de Salbutamol	Broncodilatador
8	9	Prednisona	Corticosteroide
9	10	Omeprazol	Inibidor da bomba de prótons
10	1	Paracetamol	Antipirético e analgésico
11	11	Doce de manga	Antipirético e analgésico

```

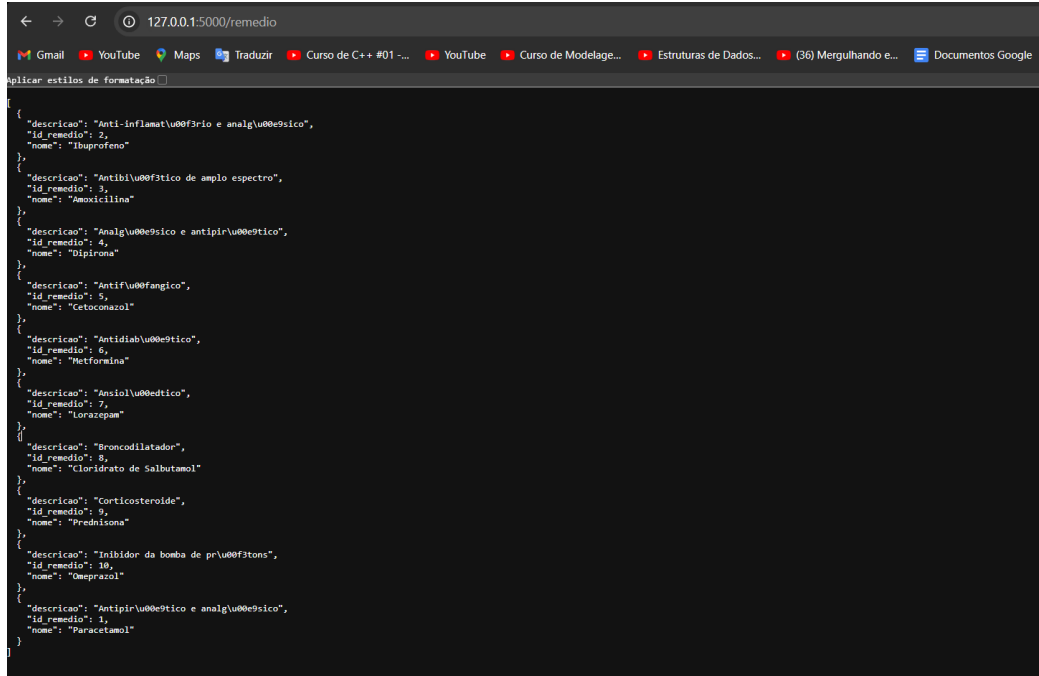
@app.route('/remedio/<int:id>', methods=['PUT'])
def atualizar_remedio(id):
    conn = get_db_connection()
    if conn is None:
        return jsonify({"error": "Não foi possível conectar ao banco de dados"})

    data = request.json
    nome = data.get('nome')
    descricao = data.get('descricao')

    try:
        with conn.cursor() as cur:
            if nome:
                cur.execute("UPDATE farmacia.remedio SET nome = %s WHERE id_remed= %s")
            if descricao:
                cur.execute("UPDATE farmacia.remedio SET descricao = %s WHERE id_remed= %s")
            conn.commit()
            return jsonify({"status": "Remédio atualizado com sucesso"})
    except Exception as e:
        return jsonify({"error": str(e)}), 500
    finally:
        conn.close()

```

## 14.8 Após usar o UPDATE

A screenshot of a web browser window with the address bar showing '127.0.0.1:5000/remedio'. The browser displays a JSON array of medicine records. The records include fields for 'descricao' (description), 'id\_remedio' (ID), and 'nome' (name). The medicines listed are Ibuprofeno, Amoxicilina, Diprofona, Cetocanazol, Metformina, Lorazepam, Cloridrato de Salbutamol, Prednisona, Omeprazol, and Paracetamol.

```
{
  "descricao": "Anti-inflamat\u00f3rio e analg\u00e9sico",
  "id_remedio": 2,
  "nome": "Ibuprofeno"
},
{
  "descricao": "Antib\u00edotico de amplo espectro",
  "id_remedio": 3,
  "nome": "Amoxicilina"
},
{
  "descricao": "Analg\u00e9sico e antipir\u00e9tico",
  "id_remedio": 4,
  "nome": "Diprofona"
},
{
  "descricao": "Antif\u00fongico",
  "id_remedio": 5,
  "nome": "Cetocanazol"
},
{
  "descricao": "Antidiab\u00e9tico",
  "id_remedio": 6,
  "nome": "Metformina"
},
{
  "descricao": "Ansio\u00e9tico",
  "id_remedio": 7,
  "nome": "Lorazepam"
},
{
  "descricao": "Broncodilatador",
  "id_remedio": 8,
  "nome": "Cloridrato de Salbutamol"
},
{
  "descricao": "Corticoester\u00f3ide",
  "id_remedio": 9,
  "nome": "Prednisona"
},
{
  "descricao": "Inibidor da bomba de pr\u00f3tons",
  "id_remedio": 10,
  "nome": "Omeprazol"
},
{
  "descricao": "Antipir\u00e9tico e analg\u00e9sico",
  "id_remedio": 1,
  "nome": "Paracetamol"
}
```

## 14.9 Delete (Excluir)

Para excluir um registro, voc\u00ea usa a opera\u00e7\u00e3o DELETE. Aqui est\u00e1 um exemplo para excluir um rem\u00e9dio:

```
@app.route('/remedio/<int:id>', methods=['DELETE'])
def excluir_remedio(id):
    conn = get_db_connection()
    if conn is None:
        return jsonify({"error": "N\u00e3o foi poss\u00edvel conectar ao banco de dados"})

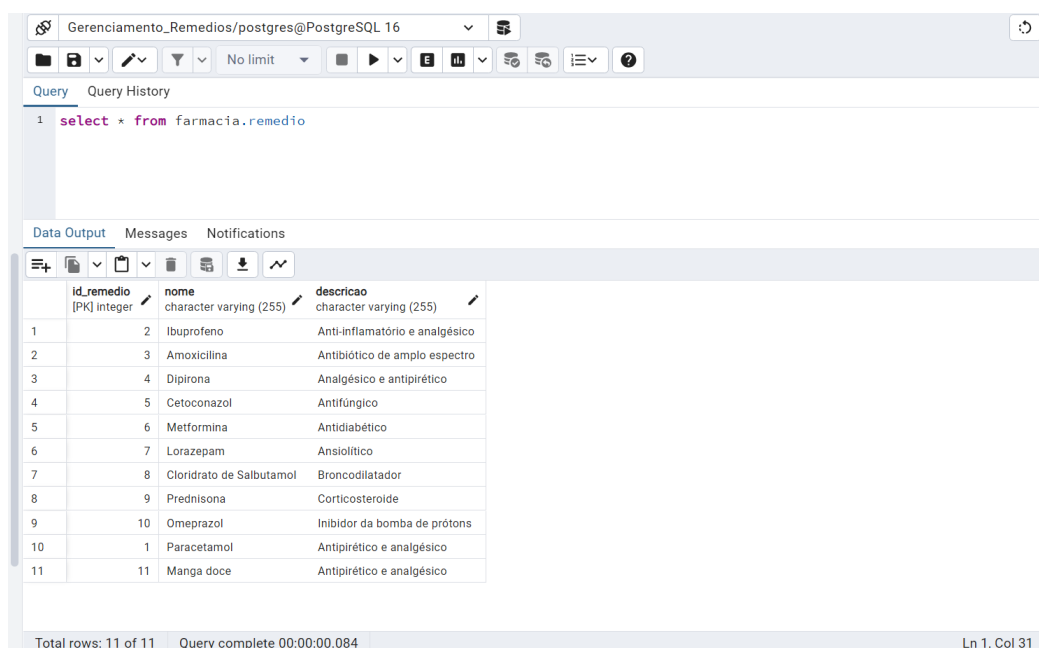
    try:
        with conn.cursor() as cur:
            cur.execute("DELETE FROM farmacia.remedio WHERE id_remedio = %s;", (
                id,
            ))
            conn.commit()
            return jsonify({"status": "Rem\u00e9dio exclu\u00eddo com sucesso"})
    except Exception as e:
        return jsonify({"error": str(e)}), 500
    finally:
        conn.close()
```

## 15 Conclusão

O sistema de gerenciamento de remédios foi implementado com sucesso, com uma API robusta e uma estrutura de banco de dados eficiente. As operações CRUD permitem um gerenciamento completo dos dados de remédios, fornecedores, e outros aspectos relacionados.

Usando o POSTMAN para gerenciar esses dados;

Com o campo "DELETE" `http://localhost:5000/remedio/11`



The screenshot shows a PostgreSQL query client interface. The query executed is `select * from farmacia.remedio`. The results are displayed in a table with 11 rows and 3 columns: `id_remedio` (integer), `nome` (character varying), and `descricao` (character varying). The table lists various medicines such as Ibuprofeno, Amoxicilina, Dipirona, and others. The status bar at the bottom indicates 'Total rows: 11 of 11' and 'Query complete 00:00:00.084'.

	<code>id_remedio</code> [PK] integer	<code>nome</code> character varying (255)	<code>descricao</code> character varying (255)
1	2	Ibuprofeno	Anti-inflamatório e analgésico
2	3	Amoxicilina	Antibiótico de amplo espectro
3	4	Dipirona	Analgésico e antipirético
4	5	Cetoconazol	Antifúngico
5	6	Metformina	Antidiabético
6	7	Lorazepam	Ansiolítico
7	8	Cloridrato de Salbutamol	Broncodilatador
8	9	Prednisona	Corticosteroide
9	10	Omeprazol	Inibidor da bomba de prótons
10	1	Paracetamol	Antipirético e analgésico
11	11	Manga doce	Antipirético e analgésico

Total rows: 11 of 11    Query complete 00:00:00.084    Ln 1, Col 31

HTTP **http://localhost:5000/remedio/11** Save Share

**DELETE** ▼ **http://localhost:5000/remedio/11** Send ▼

Params Auth Headers (8) **Body** • Scripts Tests Settings Cookies

raw ▼ **JSON** ▼ Beautify

```
1 {
2   ... "descricao": "Antipirético e analgésico",
3   ... "id_remedio": 11,
4   ... "nome": "Doce de manga"
5 }
```

Body ▼ 200 OK 76 ms 217 B Save as example ...

Pretty Raw Preview Visualize **JSON** ▼ ≡ 📄 🔍

```
1 {
2   "message": "Operação bem-sucedida!"
3 }
```

Gerenciamento\_Medios/postgres@PostgreSQL 16

**Query** Query History

```
1 select * from farmacia.remedio
```

Data Output Messages Notifications

	id_remédio [PK] integer	nome character varying (255)	descricao character varying (255)
1	2	Ibuprofeno	Anti-inflamatório e analgésico
2	3	Amoxicilina	Antibiótico de amplo espectro
3	4	Dipirona	Analgésico e antipirético
4	5	Cetoconazol	Antifúngico
5	6	Metformina	Antidiabético
6	7	Lorazepam	Ansiolítico
7	8	Cloridrato de Salbutamol	Broncodilatador
8	9	Prednisona	Corticosteroide
9	10	Omeprazol	Inibidor da bomba de prótons
10	1	Paracetamol	Antipirético e analgésico

<b>Tarefa</b>	<b>Quem Realizou</b>
Implementação de Atributos Multivalorados	Kaio
Implementação da Hierarquia de Herança	Kaio
Prototipagem do SQL do banco de dados	Kaio
Modelo Relacional	Tailan de Souza
Modelagem e Implementação do Banco de Dados Final	Tailan de Souza
Popular tabelas	Tailan de Souza
Testar Aplicação	Tailan de Souza
Desenvolvimento e Configuração do Projeto	Tailan de Souza
Criação de Documentos e Relatórios	Tailan de Souza
Trabalho com GitHub	Tailan de Souza
CRUD da Aplicação	Tailan de Souza
Revisão Geral	Tailan de Souza

## 16 Referências

### Referências

- [1] Elmasri, R., & Navathe, S. B. (2024). *Fundamentals of Database Systems*. Editora ABC.

## 17 Link do Repositório

[https://github.com/Naliat/trabalho\\_fbd](https://github.com/Naliat/trabalho_fbd)