



**CENTRO UNIVERSITÁRIO INTERNACIONAL UNINTER  
ESCOLA SUPERIOR POLITÉCNICA**

**PROJETO “SGHSS” (SISTEMA DE GESTÃO HOSPITALAR E DE SERVIÇOS DE SAÚDE).**

**ATIVIDADE PRÁTICA**

**LUIZ CARLOS REZENDE, 4303616**

**SANTA RITA DO SAPUCAÍ-MG  
2025**

## **1 PLANEJAMENTO E COMPREENSÃO Leitura Aprofundada do Estudo de Caso.**

A leitura aprofundada do estudo de caso referente ao Sistema de Gestão Hospitalar e de Serviços de Saúde (SGHSS) tem como finalidade compreender os desafios, requisitos e impactos relacionados à implementação de um sistema informatizado na gestão hospitalar. Essa etapa é fundamental para assegurar que as necessidades dos usuários sejam contempladas e que o sistema seja desenvolvido de forma eficiente e eficaz.

O principal objetivo desta análise é identificar os problemas enfrentados pelas instituições de saúde e avaliar como um sistema de gestão hospitalar pode contribuir para a melhoria da prestação dos serviços. A investigação abrange os seguintes aspectos:

**1-1 Eficiência operacional:** redução de erros administrativos e otimização dos processos internos;

**1-2 Atendimento ao paciente:** melhoria na comunicação entre setores e maior agilidade no atendimento;

**1-3 Segurança da informação:** proteção dos dados dos pacientes, em conformidade com as normas regulatórias vigentes.

### **1.2 Contextualização e Problemática**

O setor da saúde enfrenta desafios significativos no que se refere à gestão de informações, tais como:

1-Fragmentação dos dados entre os diversos setores do hospital;

2-Ausência de padronização nos registros eletrônicos;

3-Lentidão no acesso às informações essenciais para a tomada de decisões;

4-Necessidade de conformidade com normas legais de segurança e privacidade de dados.

Para a realização de uma análise mais abrangente, foram utilizadas fontes como relatórios institucionais, entrevistas com profissionais da área da saúde, estudos de caso de sistemas similares e revisão da literatura acadêmica.

A presente análise envolveu:

1=Levantamento das necessidades das instituições hospitalares;

2=Identificação de problemas recorrentes na gestão hospitalar sem o uso de tecnologias;

3=Análise de soluções tecnológicas existentes e avaliação de seus impactos;

4=Avaliação dos benefícios e desafios relacionados à implementação do SGHSS.

Os resultados demonstram que a ausência de um sistema integrado contribui para atrasos operacionais, desperdício de recursos e falhas de comunicação entre setores. Por outro lado, a adoção de um sistema digital pode trazer benefícios relevantes, como:

A=Redução no tempo de espera dos pacientes;

B=Automação de processos administrativos;

C=Conformidade com normas de segurança e privacidade de dados.

## **2 DEFINIÇÃO DO ESCOPO**

### **2.1 Objetivo do Projeto**

O Sistema de Gestão Hospitalar e de Serviços de Saúde (SGHSS) tem como objetivo principal desenvolver um sistema informatizado para apoiar a gestão hospitalar, promovendo a otimização de processos administrativos, a melhoria da qualidade do atendimento aos pacientes e a garantia de segurança da informação, conforme as normas regulatórias, como a Lei Geral de Proteção de Dados (LFPD).

### **2.2 Escopo Funcional**

O escopo funcional do SGHSS contempla as seguintes funcionalidades principais:

**Gestão de Pacientes:** Cadastro de pacientes com informações obrigatórias (como CPF e e-mail), atualização de dados e armazenamento seguro das informações.

1=Caso de uso: cadastrar paciente com dados válidos → exibir mensagem “Paciente cadastrado com sucesso”.

2=Caso de uso: tentar cadastrar paciente sem informar o CPF → exibir mensagem de erro e impedir o cadastro.

3=Funcionalidade adicional: autenticação de pacientes no sistema.

**4=Agendamento de Consultas e Procedimentos:** Controle de agenda para marcação de consultas e procedimentos médicos.

**5=Segurança e Controle de Acesso:** Implementação de perfis de usuários e controle de permissões para o acesso a dados sensíveis.

**6=Segurança da Informação:** Aplicação de padrões de segurança robustos para proteção de dados pessoais e clínicos.

**7=Escalabilidade:** Estrutura do sistema preparada para atender ao crescimento da demanda hospitalar.

**8=Usabilidade:** Interface gráfica intuitiva e acessível, adequada a diferentes perfis de usuários.

**9=Alta Disponibilidade:** Garantia de funcionamento contínuo, evitando interrupções nos serviços oferecidos pela instituição de saúde.

## **Requisitos Funcionais (RF)**

Estes requisitos descrevem o que o sistema **SGHSS (Sistema de Gestão Hospitalar e de Serviços de Saúde)** deve ser capaz de realizar:

### **RF01 - Cadastrar Pessoas**

O sistema deve permitir o **cadastro de pessoas** com nome, contato e e-mail.

### **RF02 - Gerenciar Pacientes**

O sistema deve permitir:

## **RF02 – Gerenciar Pacientes**

### **RF02.1: Listar todos os pacientes**

O sistema deve retornar uma lista com todos os pacientes cadastrados no banco de dados, exibindo informações como nome, CPF, data de nascimento, etc.

### **RF02.2: Buscar pacientes por ID**

Permite consultar os dados de um paciente específico por meio de seu identificador único (ID). Útil para visualização detalhada.

### **RF02.3: Criar novo registro de paciente**

Permite cadastrar um novo paciente, fornecendo os dados obrigatórios como nome, CPF, data de nascimento, e outros campos relacionados.

### **RF02.4: Atualizar dados de um paciente**

Permite editar as informações de um paciente já existente no sistema, como atualizar telefone, endereço, etc.

### **RF02.5: Excluir um paciente do sistema**

Remove permanentemente o registro de um paciente. Esse recurso deve ser protegido para evitar exclusões acidentais.

## **RF03 – Gerenciar Doutores**

### **RF03.1: Listar todos os doutores**

Exibe todos os doutores registrados, com dados como nome, CRM, especialidade, etc.

### **RF03.2: Obter doutor por ID**

Consulta os dados de um doutor específico com base no seu identificador único.

### **RF03.3: Cadastrar um novo doutor**

Registra um novo doutor no sistema, exigindo dados como nome, especialidade, CRM, etc.

### **RF03.4: Atualizar dados de um doutor**

Permite a edição de dados do doutor, como mudança de especialidade, telefone ou atualização de cadastro.

### **RF03.5: Excluir doutor do sistema**

Remove o registro de um doutor do banco de dados. Deve ter validações, como impedir a exclusão se ele tiver atendimentos vinculados.

### **RF03.6: Associar paciente a um doutor**

Cria uma relação entre um paciente e um doutor, indicando que aquele doutor pode atender aquele paciente.

### **RF03.7: Remover associação de paciente com um doutor**

Desfaz a relação entre um doutor e um paciente, sem excluir nenhum dos registros individuais.

### **RF03.8: Associar administrador a um doutor**

Vincula um administrador a um doutor, permitindo que o administrador gerencie dados e permissões desse profissional.

### **RF03.9: Remover associação de administrador com um doutor**

Desassocia um administrador de um doutor específico, revogando a responsabilidade/gestão desse profissional.

## **RF04 – Gerenciar Administradores**

### **RF04.1: Listar todos os administradores**

Mostra todos os administradores cadastrados com seus dados principais, como nome e e-mail.

#### **RF04.2: Obter administrador por ID**

Consulta detalhada do cadastro de um administrador, acessada por meio do seu ID.

#### **RF04.3: Cadastrar novo administrador com senha e e-mail**

Criação de um novo perfil de administrador, exigindo e-mail único e senha segura (com criptografia).

#### **RF04.4: Atualizar dados do administrador**

Permite alterar informações do administrador, como e-mail, nome ou redefinir senha.

#### **RF04.5: Excluir administrador**

Remove um administrador do sistema. Também deve verificar se há vínculos antes da exclusão.

### **RF05 – Relacionamentos e Atendimentos**

#### **RF05.1: O sistema deve permitir que um doutor atenda a vários pacientes**

Implementa a relação de **um para muitos** entre doutor e pacientes. Ou seja, um único doutor pode estar vinculado a diversos pacientes.

#### **RF05.2: O sistema deve permitir que vários administradores sejam associados a vários doutores**

Implementa uma relação **muitos para muitos** entre administradores e doutores, permitindo a gestão compartilhada entre diferentes administradores.

### **Requisitos Não Funcionais (RNF)**



Estes requisitos definem **como** o sistema deve se comportar em termos de desempenho, segurança, qualidade, etc.

#### **RNF 01 - Segurança da Informação**

RNF 01.1: As senhas de administradores devem ser armazenadas de forma criptografada.

RNF 01.2: O sistema deve implementar autenticação e controle de acesso para proteger os dados dos usuários.

#### **RNF 02 - Usabilidade**

RNF 02.1: A interface do sistema deve ser intuitiva e fácil de usar, permitindo navegação clara entre funcionalidades de gerenciamento.

#### **RNF 03 - Disponibilidade**

RNF 03.1: O sistema deve estar disponível para acesso em tempo integral (24x7), salvo períodos de manutenção programada.

#### **RNF 04 - Manutenibilidade**

RNF 04.1: O sistema deve ser desenvolvido em arquitetura modular (seguindo o padrão MVC) para facilitar a manutenção e evolução futura.

#### **RNF 05 - Compatibilidade**

RNF 05.1: O sistema deve ser compatível com navegadores modernos e acessíveis via web.

#### **RNF 06 - Conformidade com LGPD**

RNF 06.1: O sistema deve estar em conformidade com a Lei Geral de Proteção de Dados (LGPD), garantindo privacidade dos dados pessoais.

### 3. CRONOGRAMA

**Kanban** é uma metodologia ágil de gestão de fluxos de trabalho que visa otimizar processos e melhorar a eficiência, originalmente desenvolvida pela Toyota para a manufatura. No contexto de projetos, é utilizada por meio de um quadro visual que organiza tarefas em diferentes estágios, como "A Fazer", "Em Progresso" e "Concluído". Isso permite acompanhar o andamento das atividades e identificar gargalos, promovendo melhorias contínuas.

#### **Justificativa para o Uso**

O Kanban será utilizado no projeto para fornecer flexibilidade e visibilidade, permitindo ajustes contínuos e controle sobre o progresso das tarefas. Ele facilita a identificação rápida de problemas e gargalos, ajudando a garantir o cumprimento dos prazos.

#### **Funcionamento**

O Kanban será implementado com um quadro dividido em colunas para organizar as tarefas, limitando o número de tarefas em progresso (WIP) para evitar sobrecarga e garantir o foco nas prioridades. Esse processo ajuda na visualização clara do progresso do projeto.

Gerenciamento do SGHSS

Quadro

Filtros

Compartil

to do (a fazer)

semana 1-4 ( 23-03 a 29-03 )

Pesquisar Referências e Ferramentas

semana 1-2 ( 23-03 a 29-03 )

Definir Escopo Individual

semana 2 - 3 ( 30-03 a 05-04 )

fazer . Análise de Requisitos Detalhada

semana 2-3 ( 06-04 a 12-04 )

criar Diagramas UML e Modelos

semana 2-3 ( 06-04 a 12-04 )

Definir da Arquitetura

+ Adicionar um cartão

doing (fazendo)

semana 1-3 ( 23-03 a 29-03 )

montar Cronograma e Organização

+ Adicionar um cartão

done (feito)

semana 1-1 ( 23-03 a 29-03 )

fazer Leitura Aprofundada do Estudo de Caso

+ Adicionar um cartão

+ Adicionar outra lista

## to do (a fazer)



semana 4-6 (12-04 a 26-04 )



☐ . Desenvolvimento do Código  
(Para Foco em Back-end)



semana 7 ( 26-04 a 03-05 )

fazer Casos de Teste



semana 8 ( 04-05 a 11-05 )

. Materiais Suplementares



semana 8 ( 04 -05 a 11-05 )

Revisão e Ajustes Finais



semana 8 ( 04-05 a 11-05 )

Montagem do Documento  
Principal



+ Adicionar um cartão



#### 4. Tecnologias Utilizadas

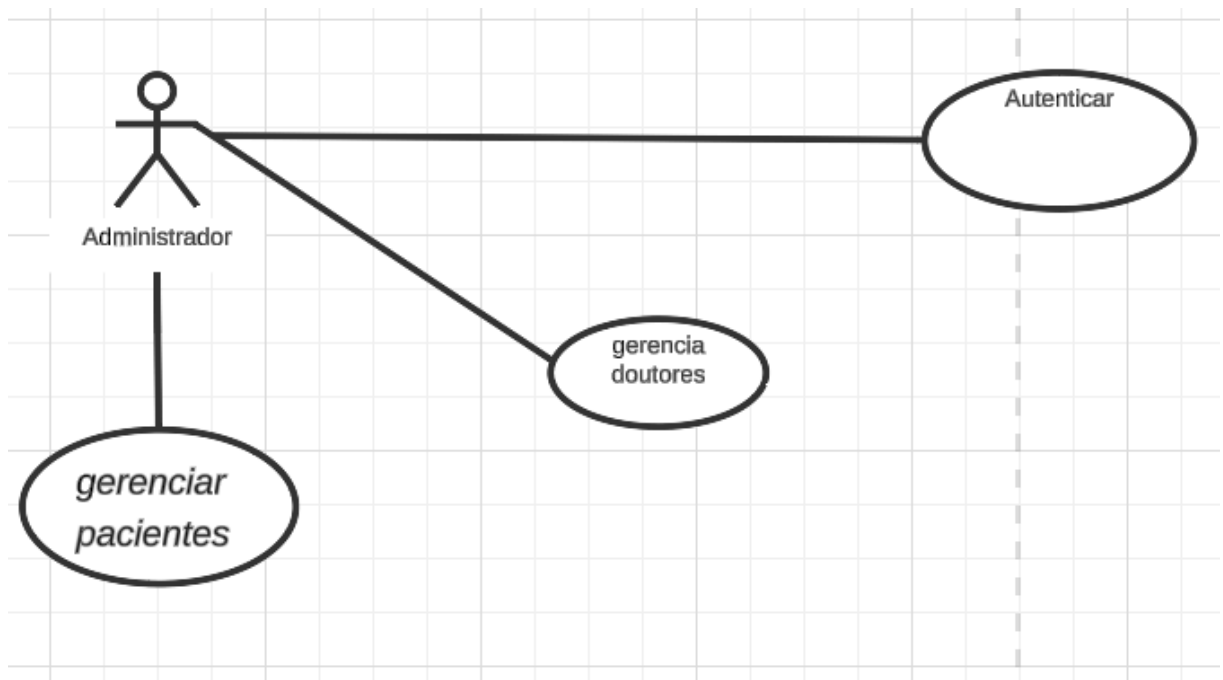
**Back-end:** NodeJs com Express.

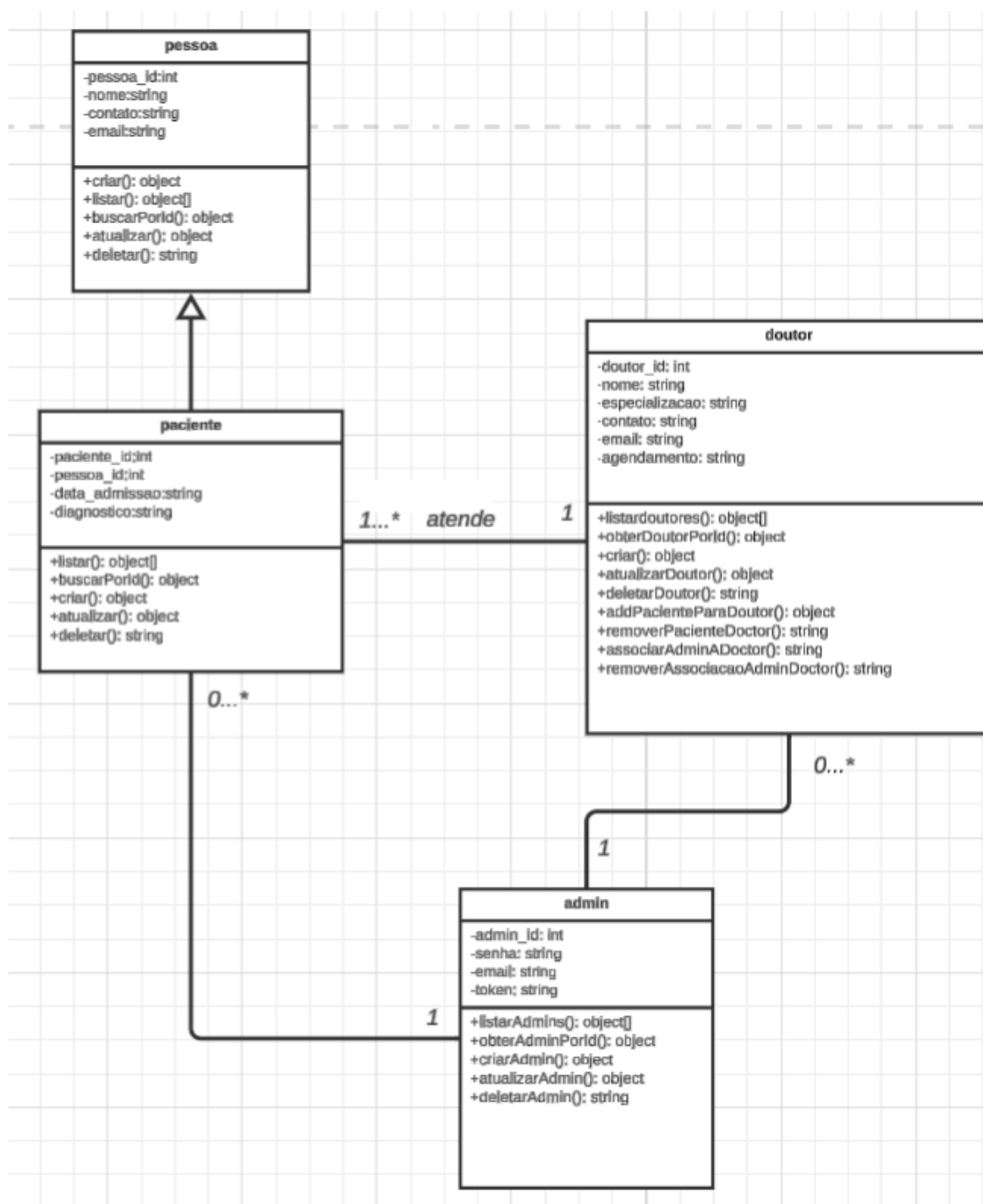
**Banco de Dados:** MySQL Workbench.

**Arquitetura:** Monolito.

**API REST:** Desenvolvimento de endpoints para integração com outros sistemas.

#### MODELAGEM E ARQUITETURA





## Descrição das Entidades – Conforme a NBR 14724:2011

### 1. Entidade: Pessoa

A entidade representa a superclasse comum a todas as demais entidades do sistema, armazenando dados genéricos de qualquer indivíduo cadastrado, sejam pacientes, doutores ou administradores. Essa entidade é base para herança de atributos.

1= **Atributos:**

A=pessoa id (int): identificador único da pessoa.

B=nome (varchar): nome completo.

C=email (varchar): endereço eletrônico único.

D=data nascimento (data): data de nascimento.

E=telefone (varchar): número de telefone.

## **2. Entidade: Paciente**

A entidade **Paciente** representa os indivíduos que utilizam os serviços de saúde da instituição. Está associada à entidade **Pessoa** por herança e armazena informações complementares relativas ao seu atendimento.

**Atributos:**

A=paciente id (int): chave primária, herda de pessoa id.

B=diagnóstico inicial (texto): livre descrevendo o estado clínico inicial do paciente.

C=admin\_id (int): chave estrangeira referenciando o administrador responsável pelo cadastro.

**Relacionamentos:**

A=Um paciente pode ser atendido por um ou mais doutores (relação de muitos para muitos).

**3. Entidade: Doutor**

A entidade **Doutor** representa os profissionais responsáveis pelo atendimento médico na instituição. Também herda da entidade **Pessoa** e agrega informações específicas da atividade médica.

**Atributos:**

A=doutor \_id (int): chave primária, herda de pessoa\_id.

B=especialidade (varchar): área médica de atuação.

C=agenda atendimento (text): horários e dias disponíveis para atendimento.

**Relacionamentos:**

A=Um doutor pode atender a vários pacientes.

B=Participa da tabela associativa **atendimento** para controle do histórico clínico.

**4. Entidade: Administrador (Admin)**



A entidade **Admin** representa os usuários do sistema responsáveis pela administração, incluindo o cadastro de pacientes e manutenção de dados.

**Atributos:**

A=\_admin (int): chave primária, herda de **pessoa\_id**.

B=email (varchar): email de acesso do administrador.

C=senha (varchar): senha do administrador (criptografada).

**Relacionamentos:**

A=Um administrador pode cadastrar múltiplos pacientes.

## **5. Entidade Associativa: Atendimento**

A entidade **Atendimento** é uma tabela associativa que registra os atendimentos realizados entre doutores e pacientes, armazenando a data e observações clínicas.

**Atributos:**

A=doutor\_id (int): chave estrangeira da entidade **Doutor**.

B=paciente\_id (int): chave estrangeira da entidade **Paciente**.

C=data\_atendimento (datetime): data e hora do atendimento.

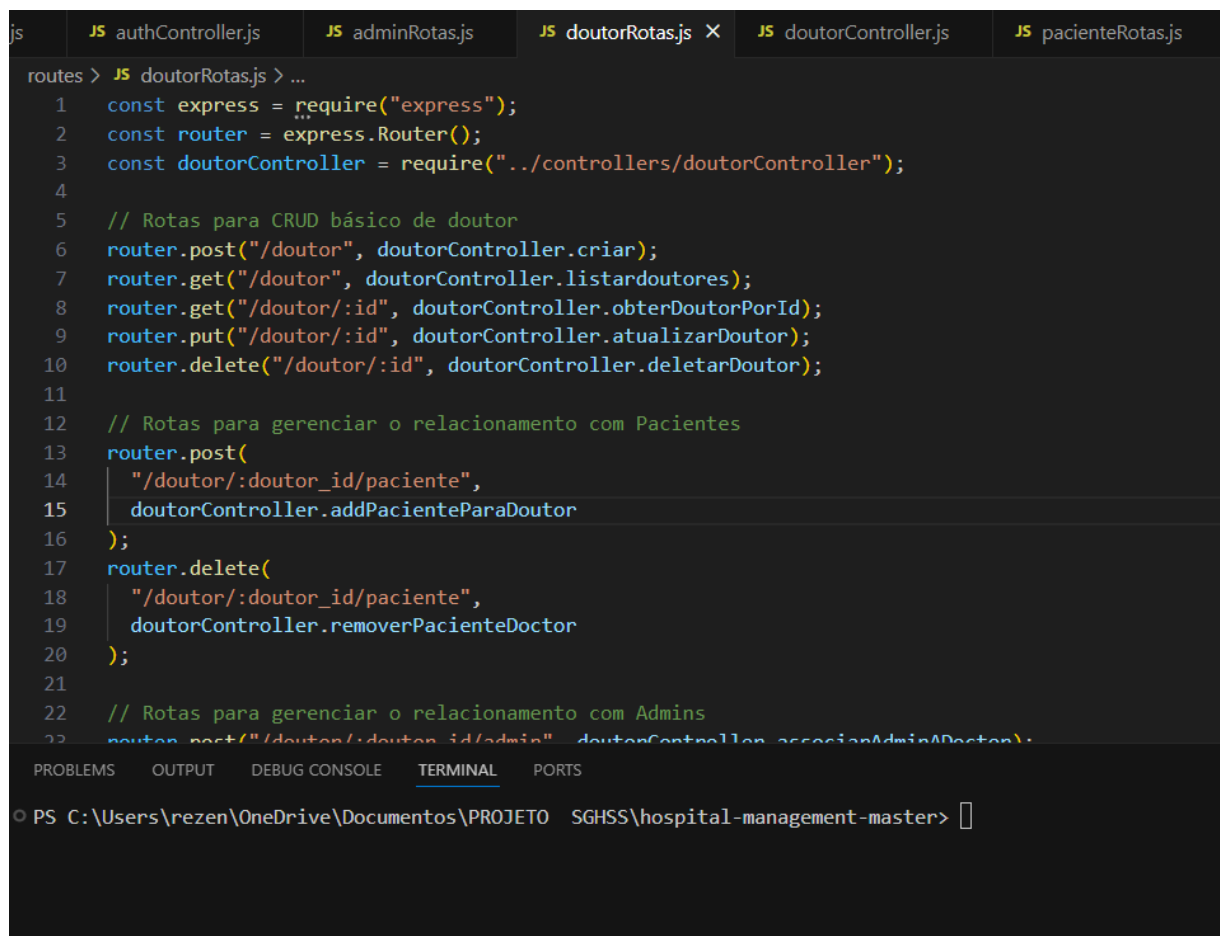
**Relacionamentos:**

A=Representa uma relação de muitos para muitos entre Paciente e Doutor.

Este modelo estabelece uma base sólida para o desenvolvimento da arquitetura da aplicação, promovendo a coesão e o reuso de código, e facilitando a manutenção e a escalabilidade do sistema SGHSS.

## IMPLEMENTAÇÃO

Esta é uma das partes mais importantes do sistema onde acontece a gestão das consultas com relacionamentos entre doutor e paciente.



```
js  JS authController.js  JS adminRotas.js  JS doutorRotas.js X  JS doutorController.js  JS pacienteRotas.js
routes > JS doutorRotas.js > ...
1  const express = require("express");
2  const router = express.Router();
3  const doutorController = require("../controllers/doutorController");
4
5  // Rotas para CRUD básico de doutor
6  router.post("/doutor", doutorController.criar);
7  router.get("/doutor", doutorController.listarDoutores);
8  router.get("/doutor/:id", doutorController.obterDoutorPorId);
9  router.put("/doutor/:id", doutorController.atualizarDoutor);
10 router.delete("/doutor/:id", doutorController.deletarDoutor);
11
12 // Rotas para gerenciar o relacionamento com Pacientes
13 router.post(
14   "/doutor/:doutor_id/paciente",
15   doutorController.addPacienteParaDoutor
16 );
17 router.delete(
18   "/doutor/:doutor_id/paciente",
19   doutorController.removerPacienteDoctor
20 );
21
22 // Rotas para gerenciar o relacionamento com Admins
23 router.post("/doutor/:doutor_id/admin", doutorController.associarAdminADoutor);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rezen\OneDrive\Documentos\PROJETO SGHSS\hospital-management-master>

## PLANO DE TESTES E QUALIDADE

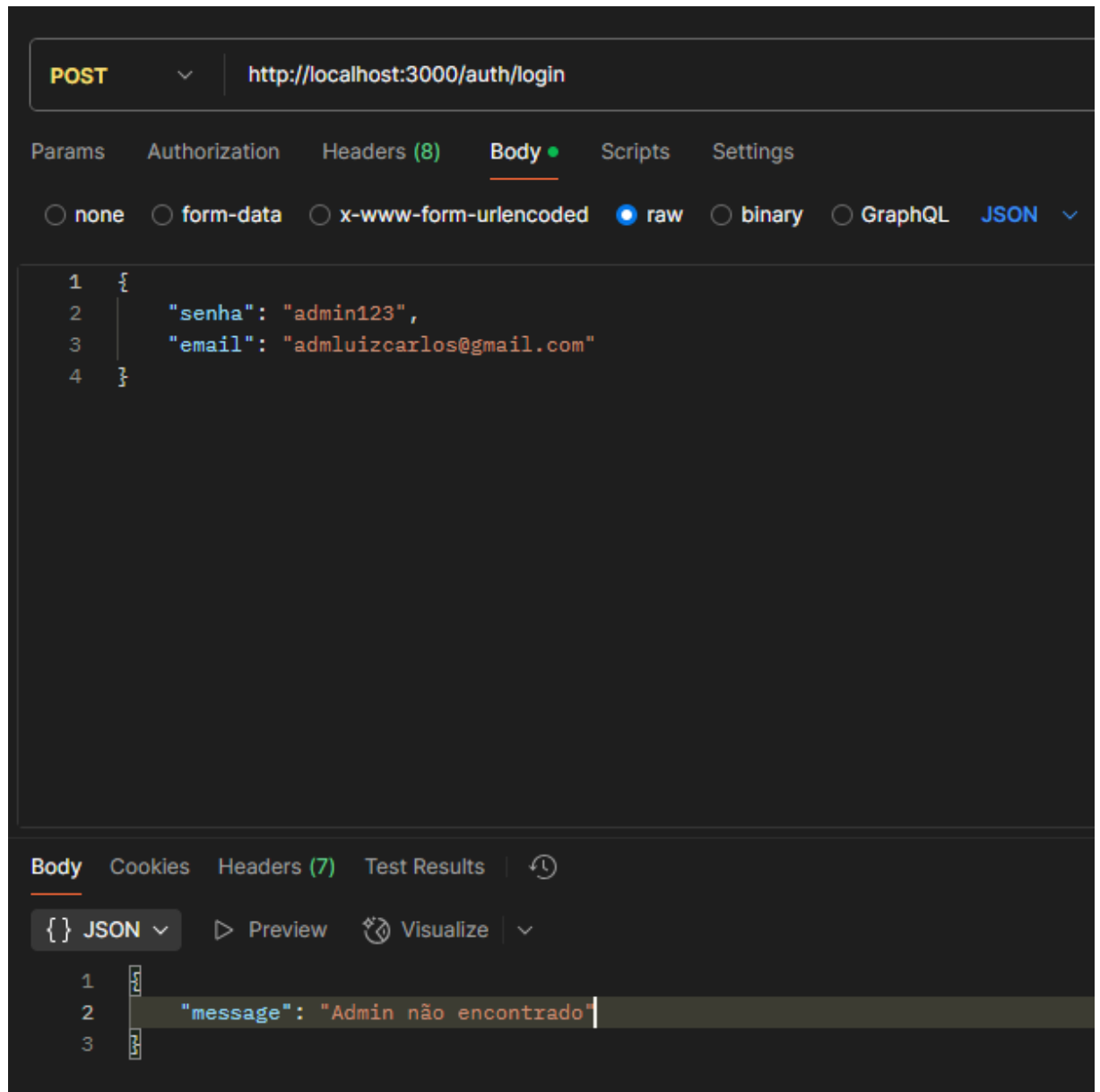
### CT001 Fazer login

#### a) Administrador não encontrado

Entrada: senha e email

Esperado: Exibir mensagem de “admin não encontrado”

Resultado: Exibição de mensagem

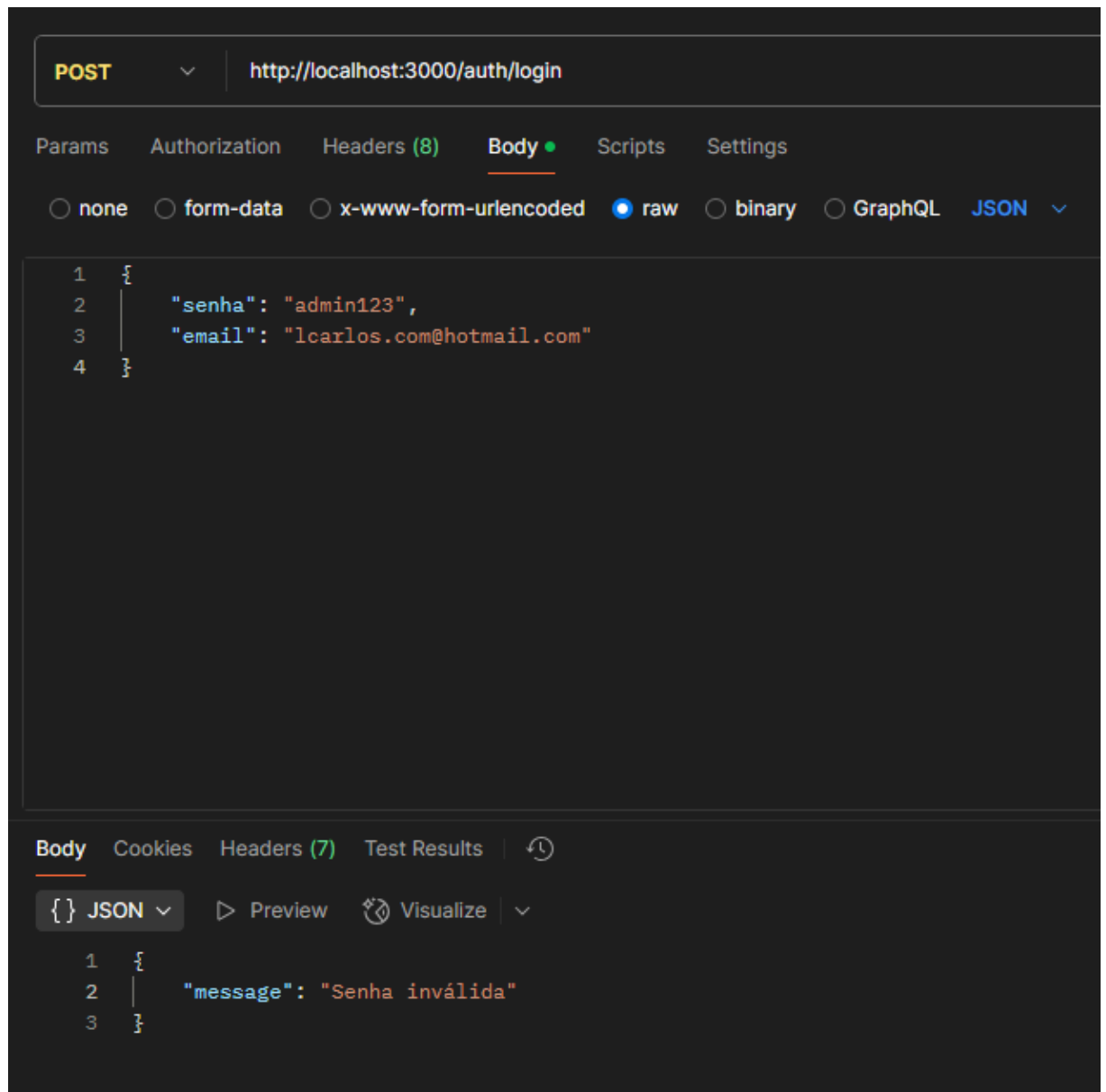


b) Senha inválida do administrador

Entrada: senha e email

Esperado: Exibir mensagem “senha inválida”

Resultado: Exibição de mensagem

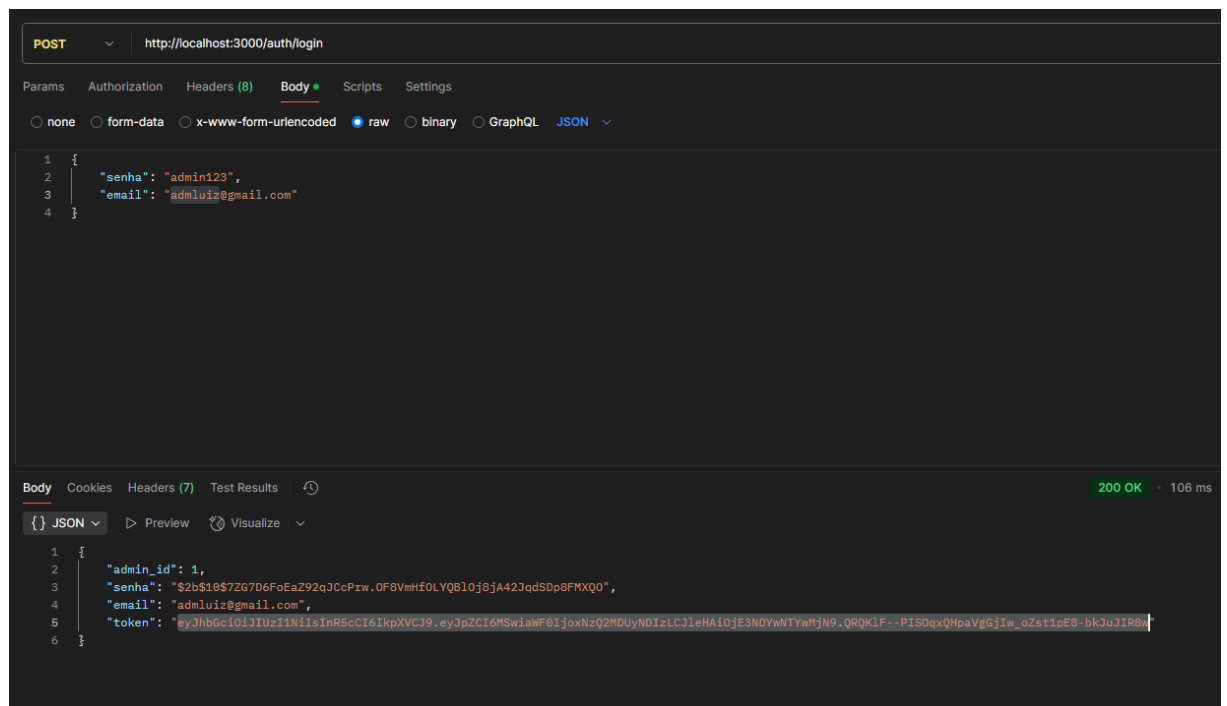


c) Login feito com sucesso deve retornar os dados do admin com token.

Entrada: senha e email

Esperado: Deve exibir senha criptografada, email e token jwt

Resultado: Exibição de mensagem

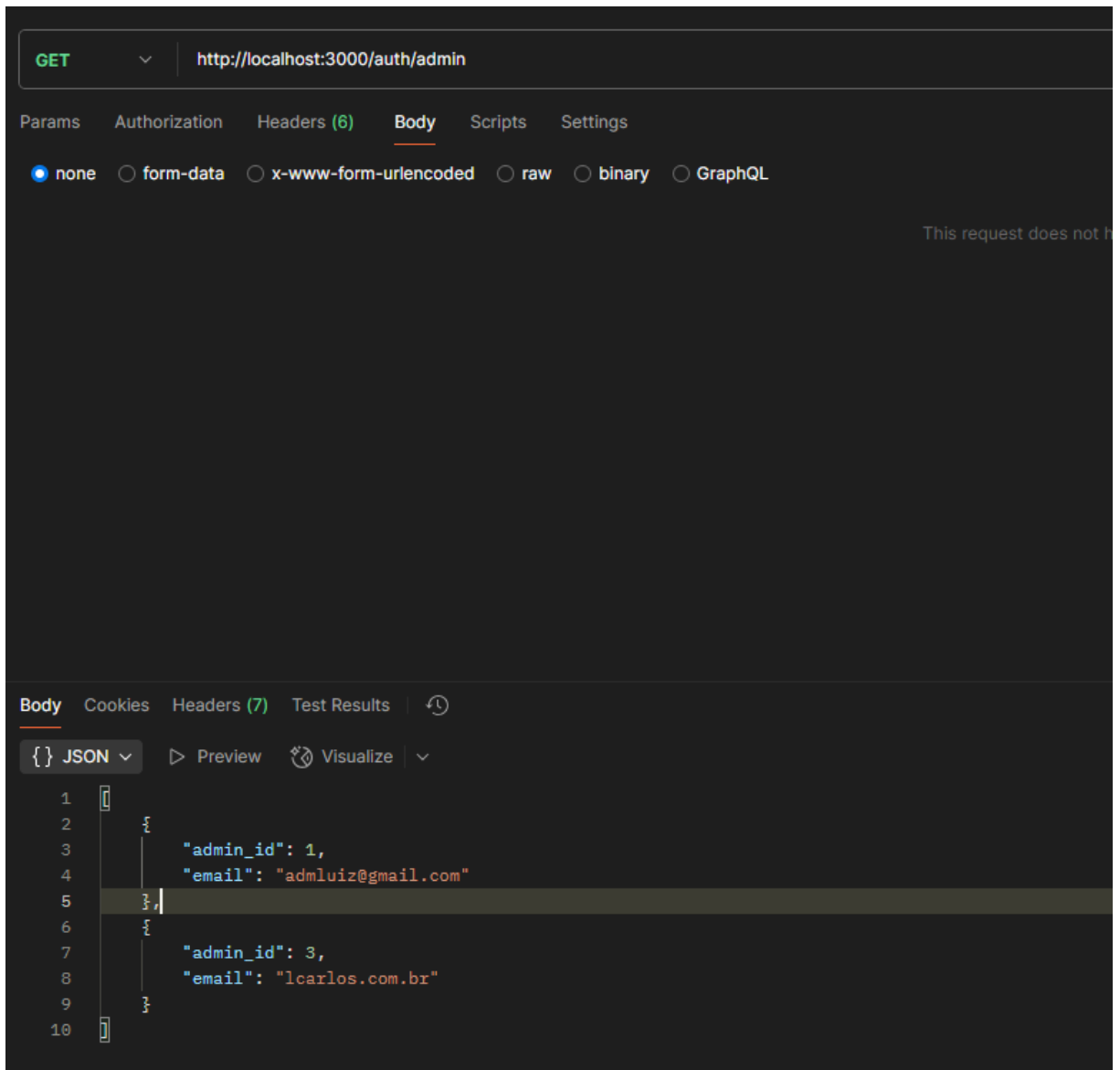


CT002 Listar todos administradores

Entrada: nenhum dado necessário

Esperado: Deve exibir lista de doutores.

Resultado: Exibição de mensagem



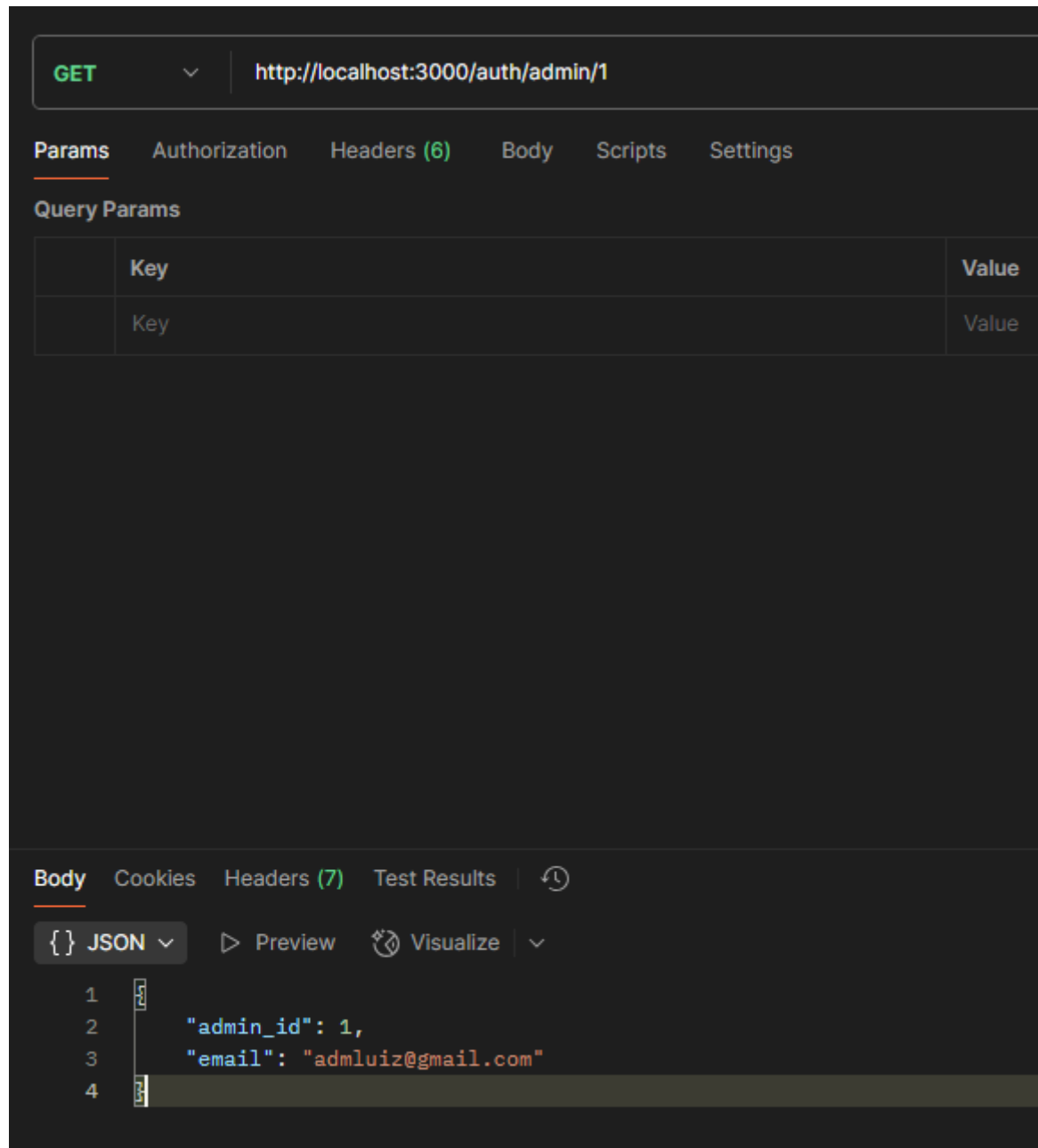
### CT003 Obter administrador por id

- a) Quando encontra o administrador o sistema deve retornar seus dados

Entrada: nenhum dado necessário

Esperado: Deve exibir “admin id :” e “email: adminluiz@gmail.com”

Resultado: Exibição de mensagem



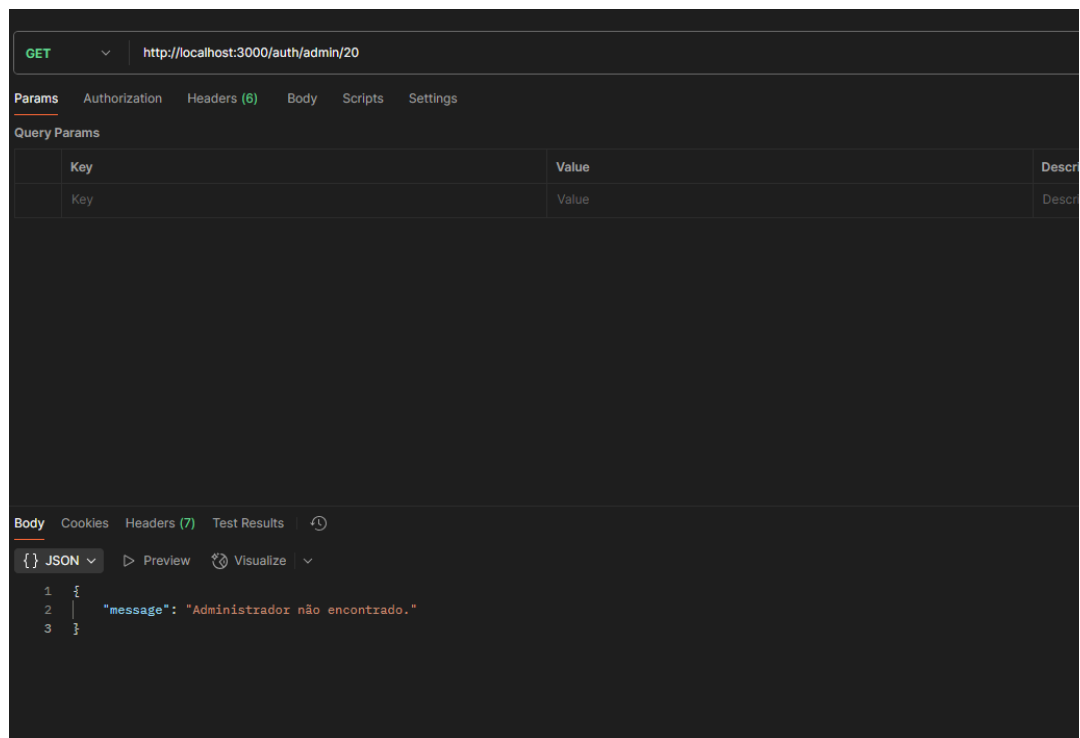
b) Caso não encontre o sistema deve retornar uma mensagem de administrador não encontrado.

Entrada: nenhum dado necessário

Esperado: Deve exibir mensagem de “administrador não encontrado”

Resultado: Exibição de mensagem





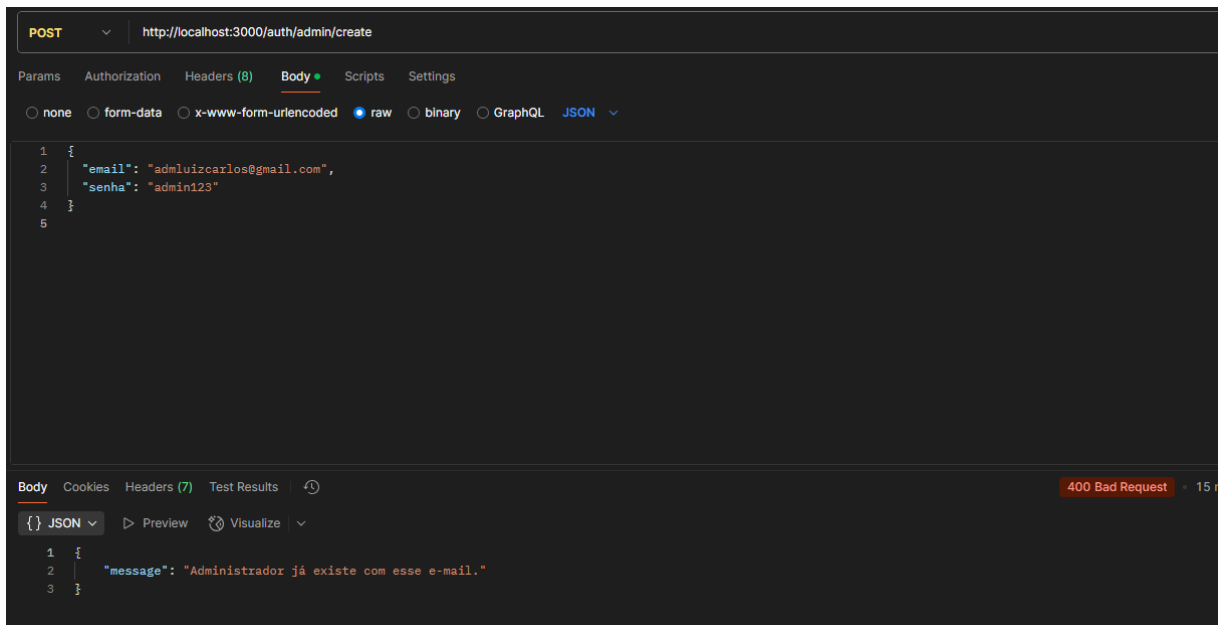
CT004 Cadastrar administrador

a) impossibilitar registro com mesmo e-mail

Entrada: senha: “admin123” email: “admluiz@gmail.com”

Esperado: Deve exibir mensagem de “administrador já existe com esse e-mail”

Resultado: Exibição de mensagem

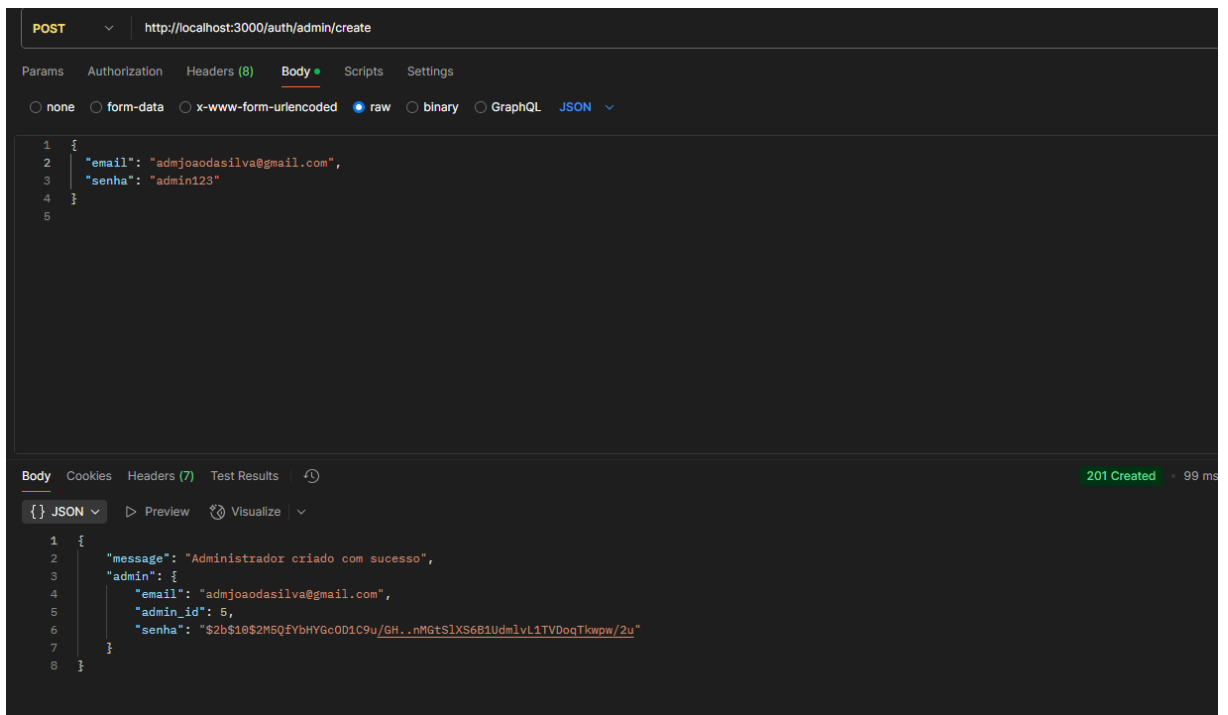


b) o sistema deve retornar a senha criptografada e a mensagem com sucesso

Entrada: senha: “admin123” email: “adminjoaodasilva@gmail.com”

Esperado: Deve exibir mensagem de “administrador criado com sucesso obtendo email e senha criptografada”.

Resultado: Exibição de mensagem



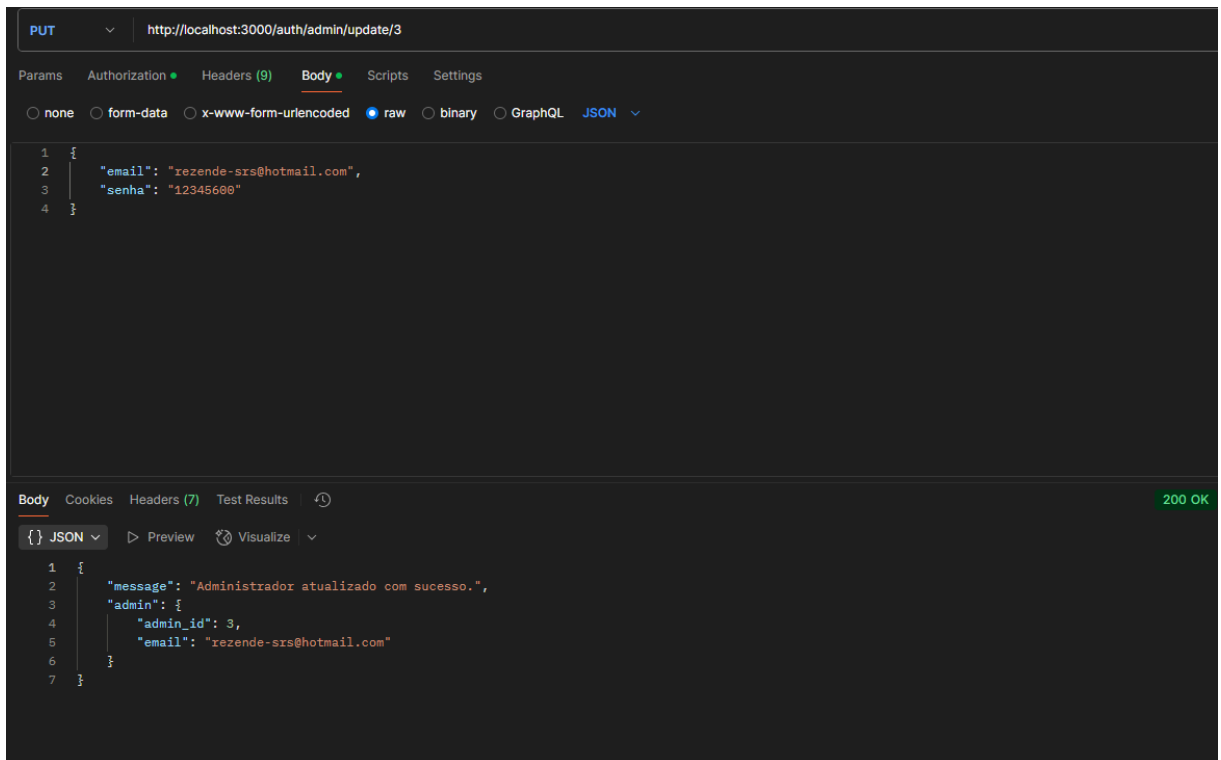
## C005 Editar administrador

A) o sistema atualiza o email do administrador

Entrada: senha: “admin123” email: “rezende-srs@hotmail.com”

Esperado: Deve exibir mensagem de “administrador atualizado com sucesso e seu email”.

Resultado: Exibição de mensagem

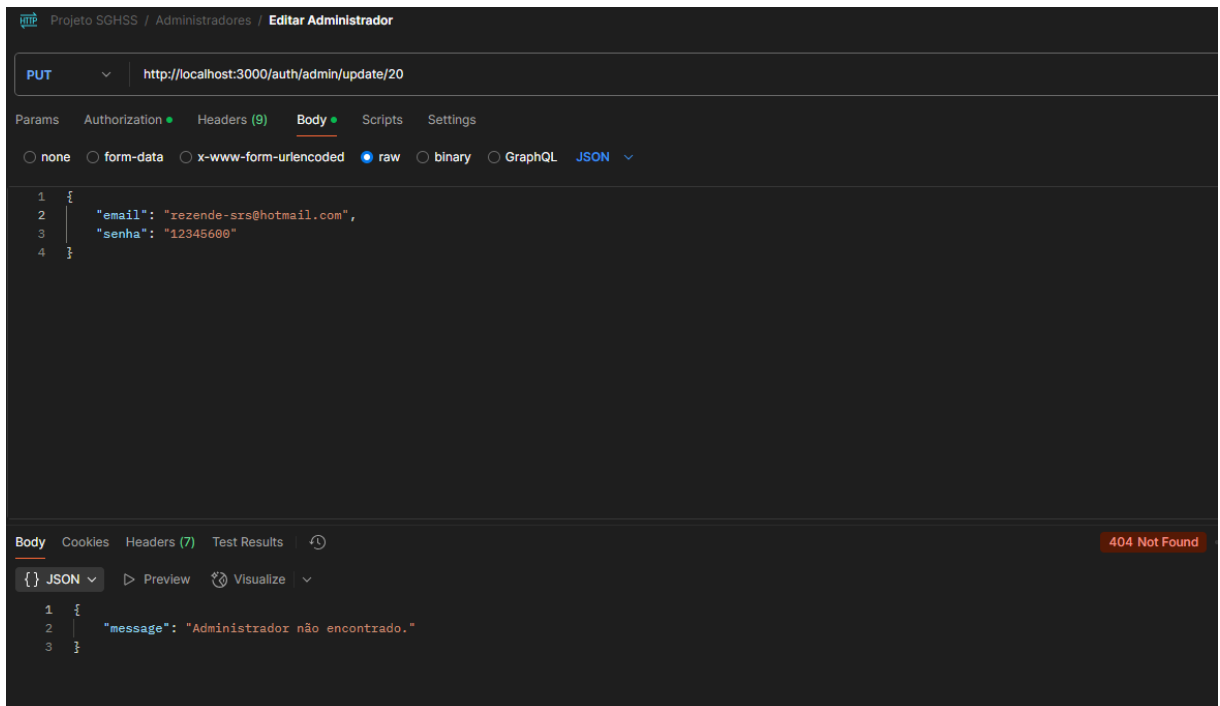


b) o sistema valida quando não encontra o administrador

Entrada: senha: “12345600” email: “rezende-srs@hotmail.com”

Esperado: Deve exibir mensagem de” administrador não encontrado”

Resultado: Exibição de mensagem



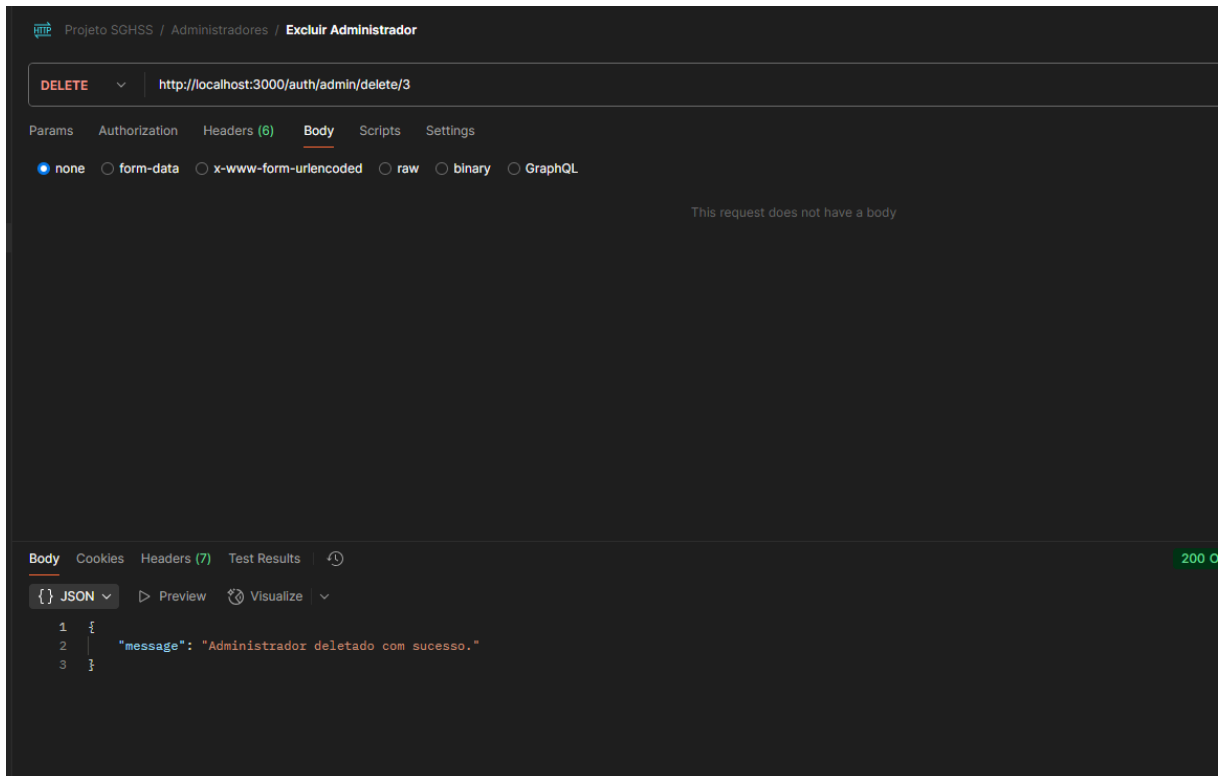
CT006 excluir administrador

a) administrador excluído com sucesso

Entrada: nenhum dado necessário

Esperado: Deve exibir mensagem de “administrado deletado com sucesso”

Resultado: Exibição de mensagem

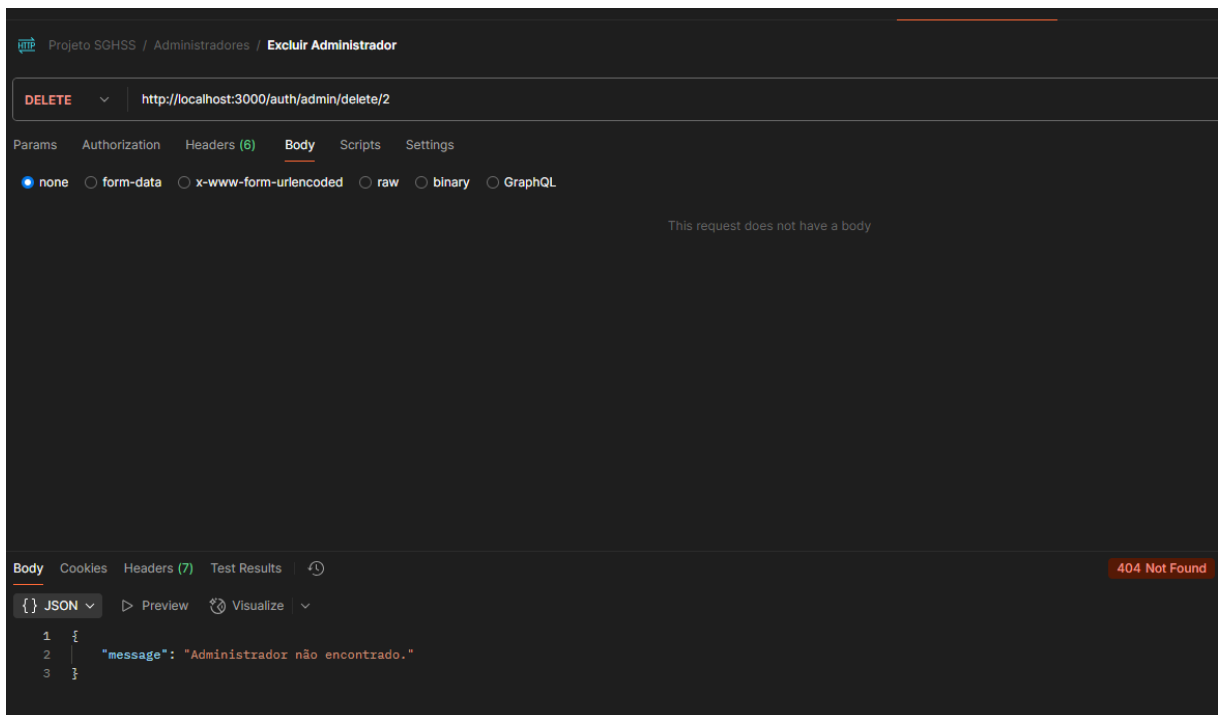


B) administrador não encontrado

Entrada: nenhum dado necessário

Esperado: Deve exibir mensagem de “administrador não encontrado”

Resultado: Exibição de mensagem



## CT007 Cadastrar paciente

Entrada: O sistema irá criar pessoas inserindo nome, contato, e-mail e data de admissão.

Esperado: Deve exibir mensagem de “pessoas criadas com sucesso” contendo nome, contato e-mail e data de admissão e mostrando os paciente criados”

Resultado: Exibição de mensagem

**POST**

http://localhost:3000/auth/pessoa

Params

Authorization ●

Headers (9)

**Body ●**

Scripts

Settings

☐ none☐ form-data☐ x-www-form-urlencoded☒ raw☐ binary☐ GraphQL

JSON

```
1  {
2    "nome": "Carla Ribeiro",
3    "contato": "11999999999",
4    "email": "carla@vidaplus.com",
5    "senha": "admin123",
6    "pessoa_id": 1,
7    "data_admissao": "01/05/2025"
8  }
9
```

**Body**

Cookies

Headers (7)

Test Results



{ } JSON ▾

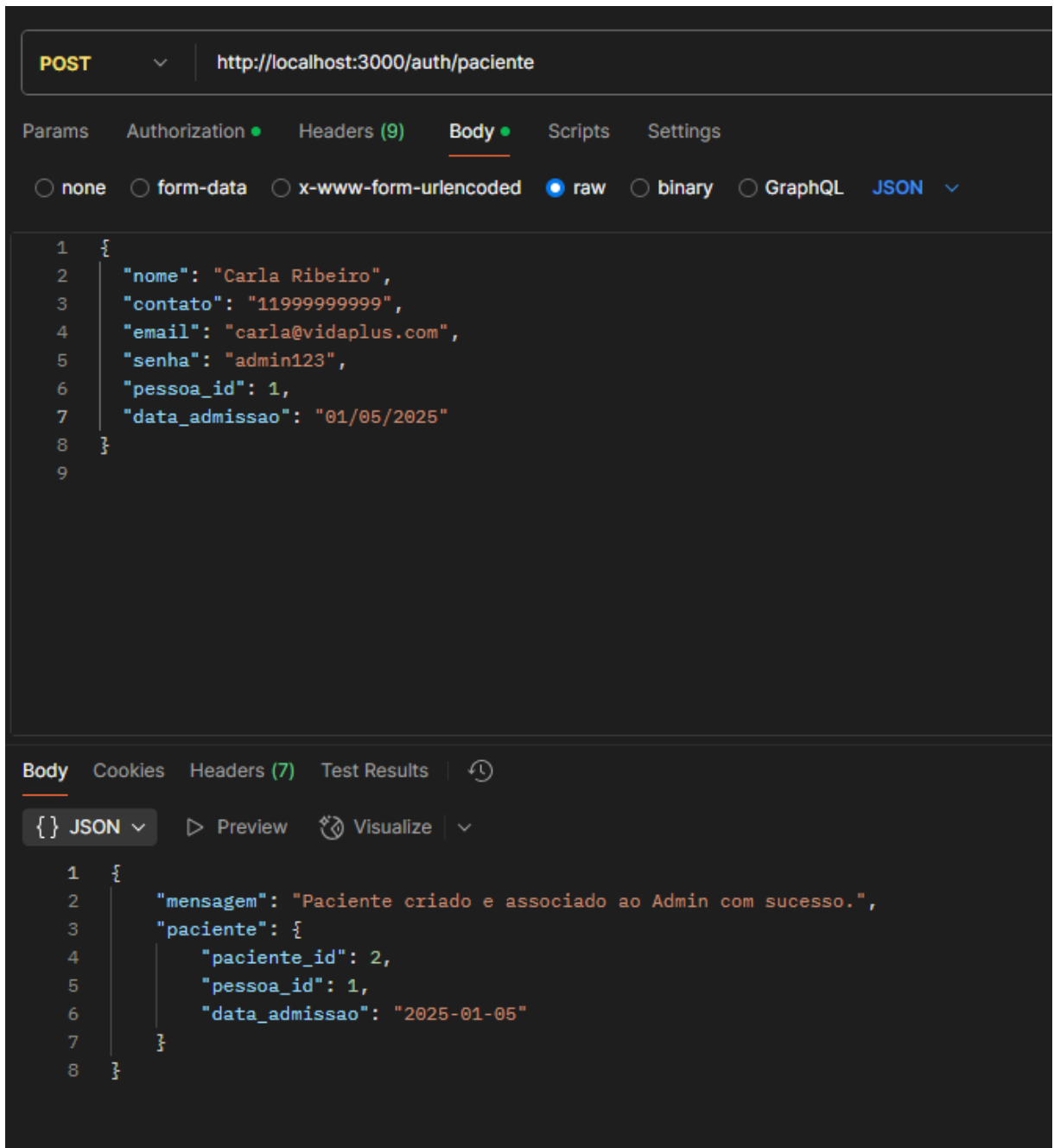
▶ Preview

Visualize



```
1  {
2    "mensagem": "Pessoa criada com sucesso.",
3    "pessoa": {
4      "pessoa_id": 1,
5      "nome": "Carla Ribeiro",
6      "contato": "11999999999",
7      "email": "carla@vidaplus.com"
8    }
9  }
```



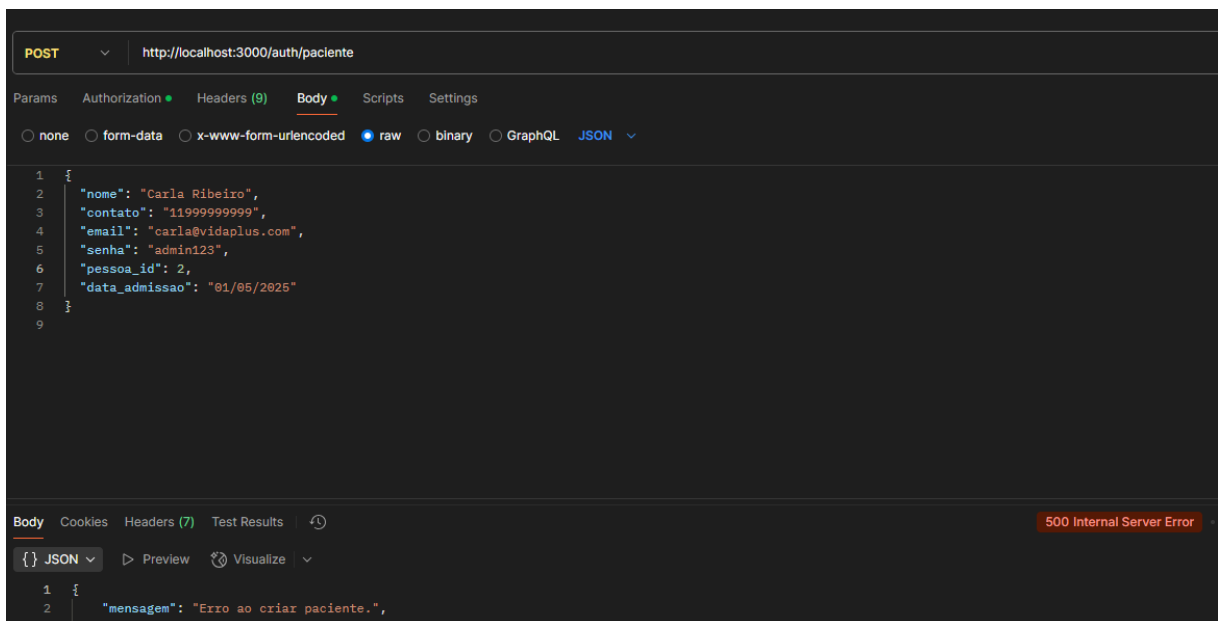


b) Mensagem de “erro ao criar paciente” quando não existe uma pessoa para associar como paciente

Entrada: O sistema irá tentar criar pessoas inserindo nome, contato, e-mail e data de admissão.

Esperado: Deve exibir mensagem de “erro ao criar paciente”

Resultado: Exibição de mensagem



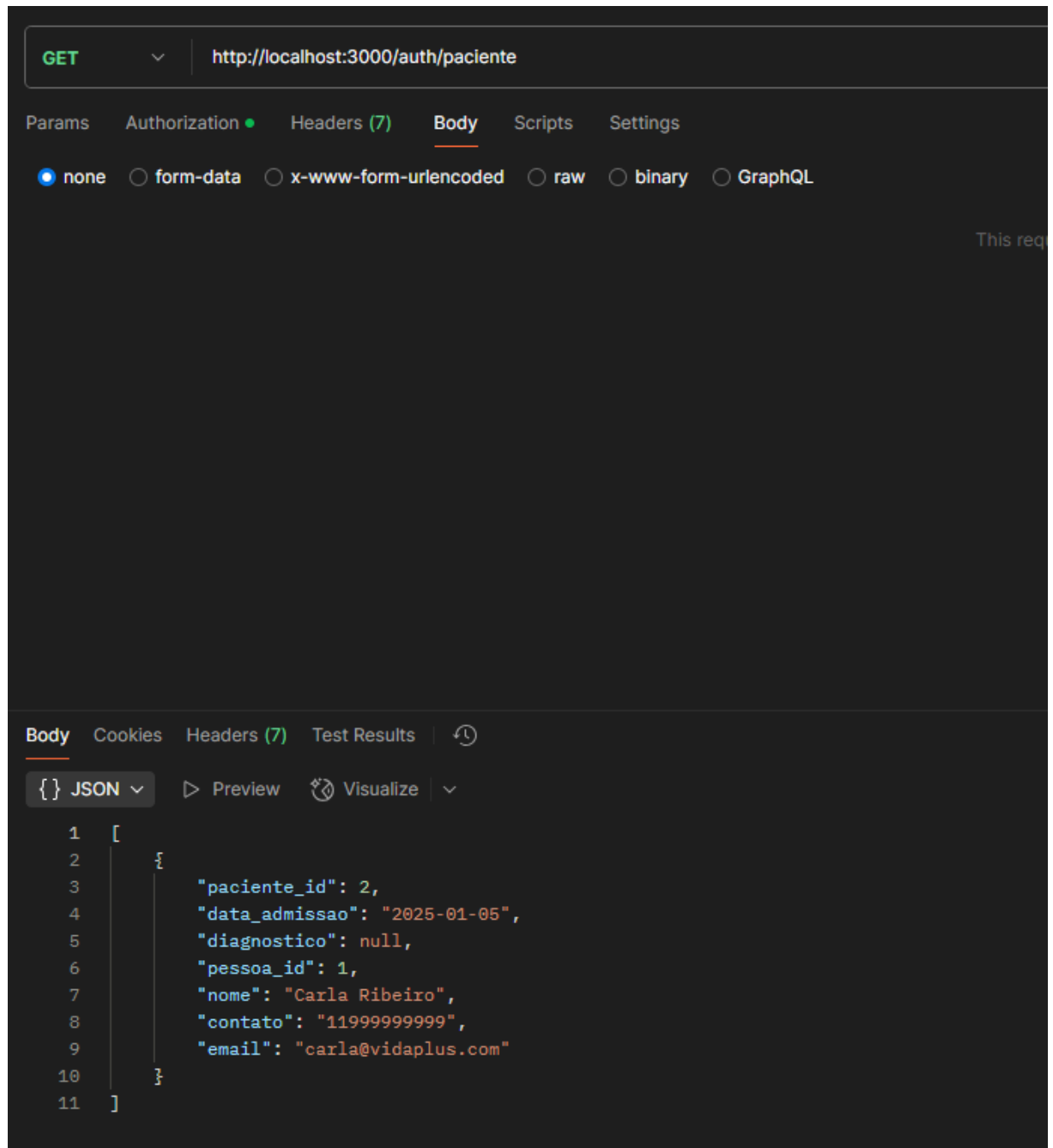
## CT008 Listar pacientes

- a) Listagem de pacientes

Entrada: nenhum dado necessário

Esperado: Deve exibir os dados dos pacientes cadastrados como: data de admissão, diagnóstico, nome, contato e e-mail.

Resultado: Lista de pacientes



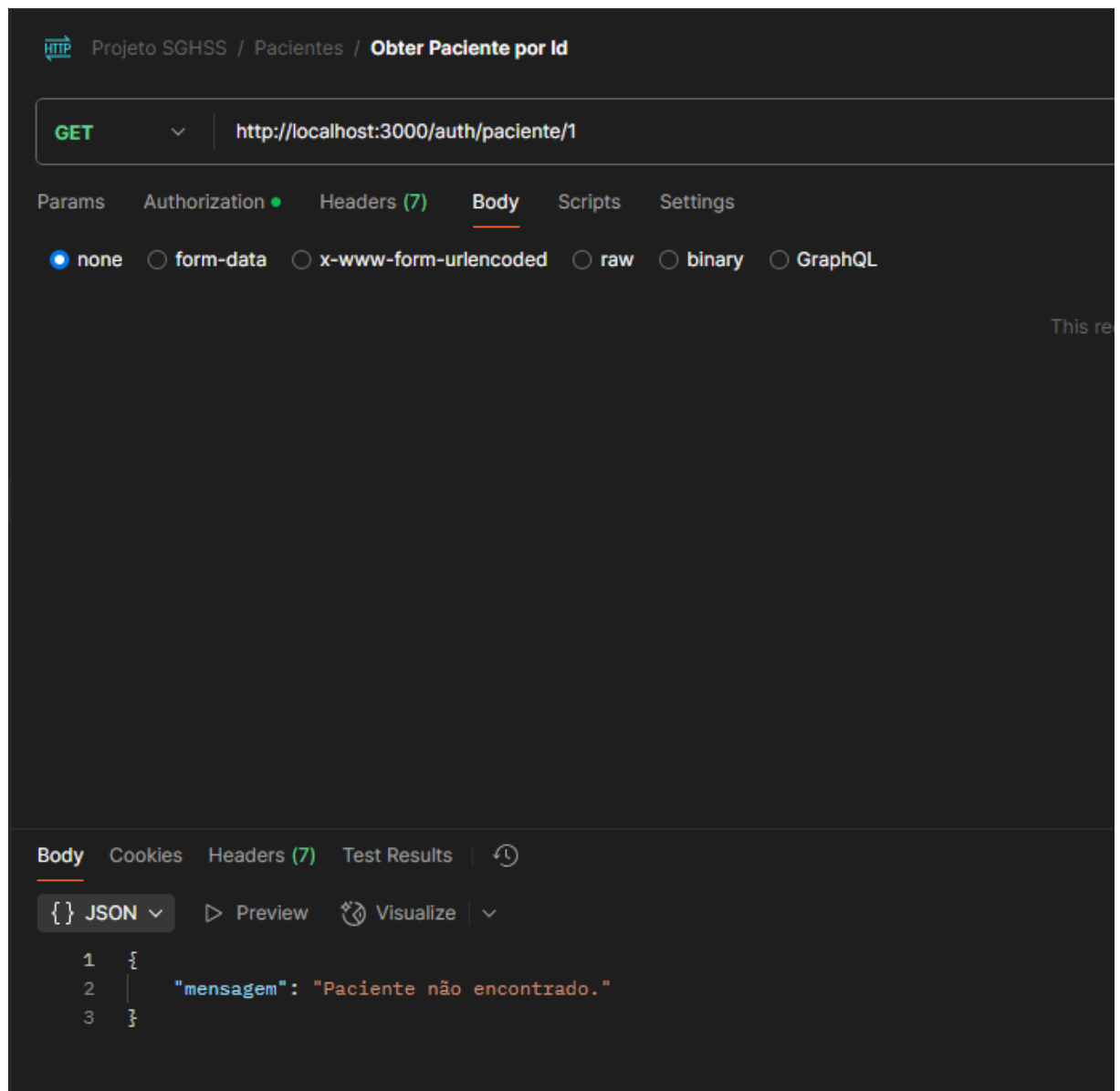
CT009 Obter paciente por id

a) Mensagem quando não encontra um paciente

Entrada: nenhum dado necessário

Esperado: Deve exibir mensagem de "Paciente não encontrado"

Resultado: Exibição de mensagem

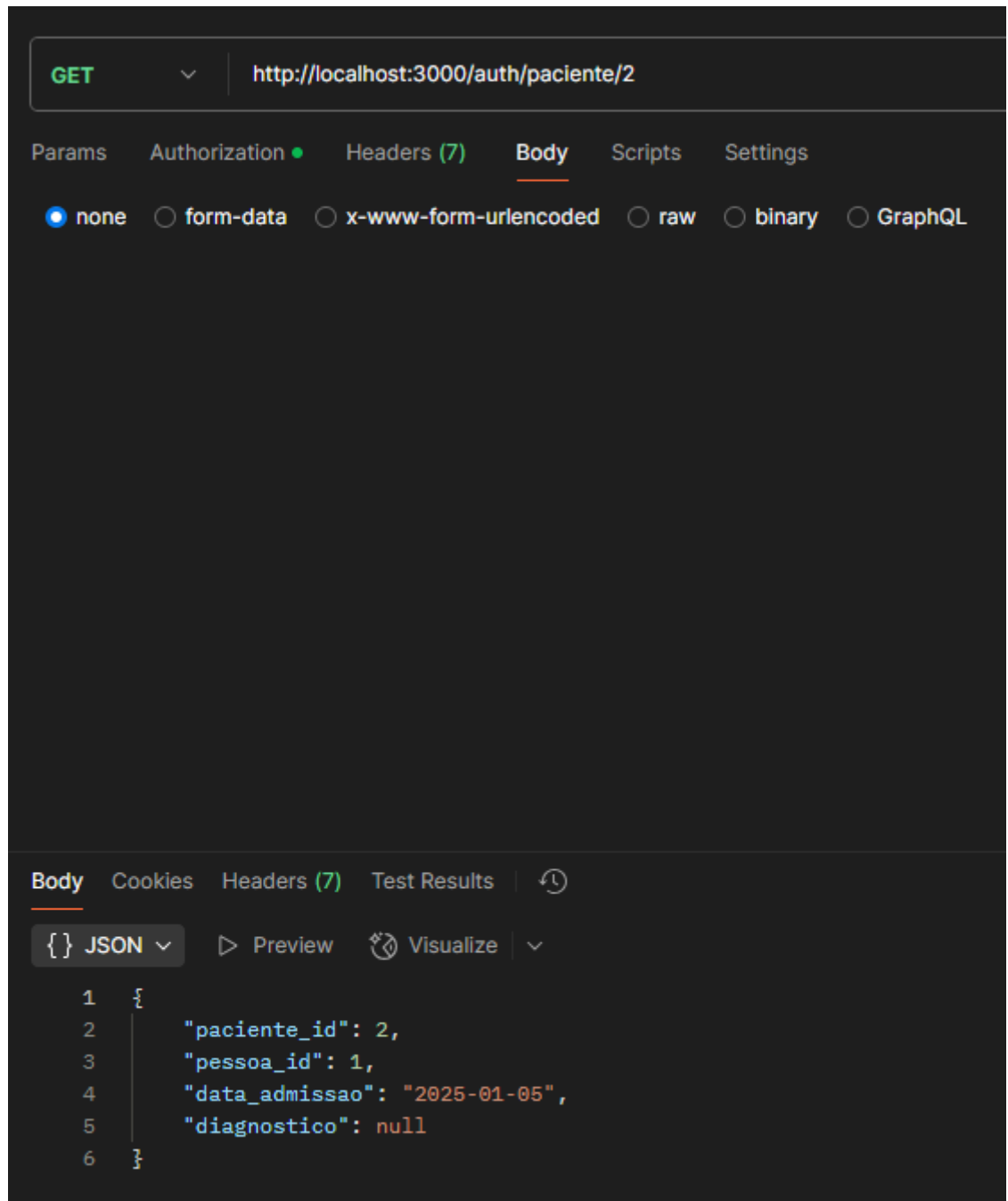


b) Quando encontra um paciente retorna os dados do paciente encontrado

Entrada: nenhum dado necessário

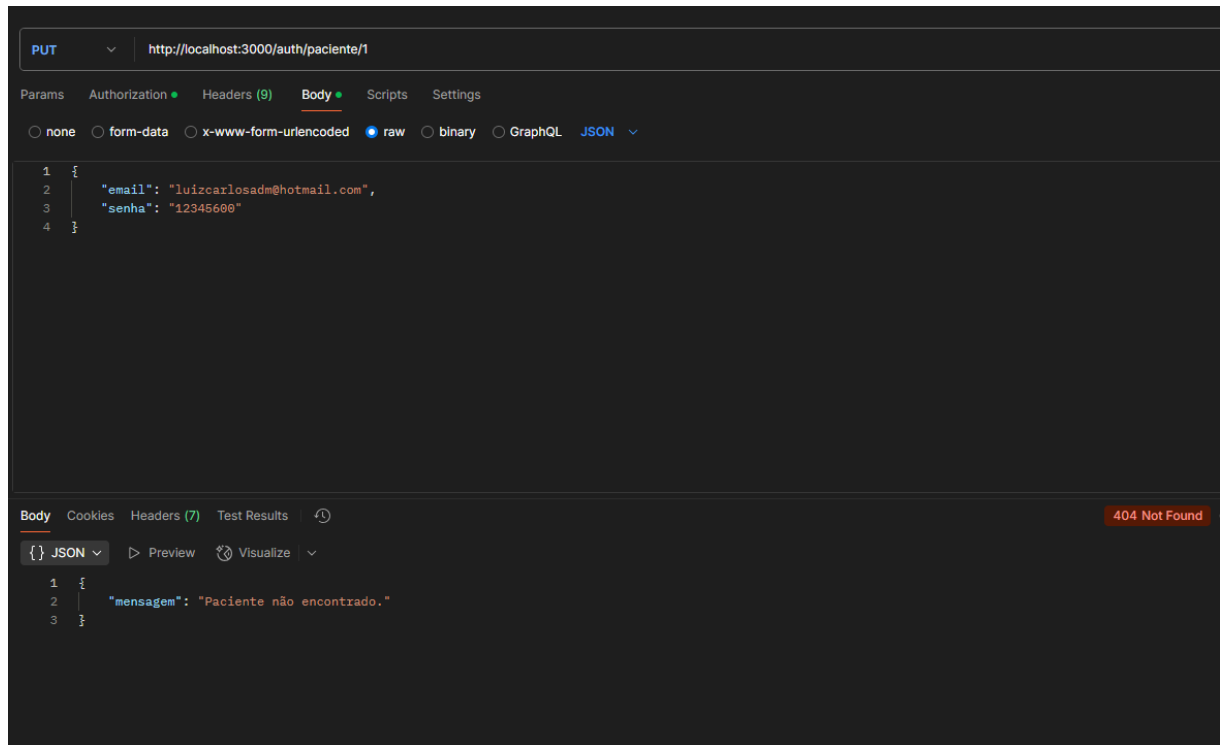
Esperado: Deve exibir mensagem de paciente, data admissão, diagnóstico.

Resultado: Exibição dos dados do paciente



#### CT010 Editar paciente

- a) Quando não encontra um paciente retorna a mensagem informando entrada:  
Entrada: senha: “12345600” e email: “luizcarlosadm@hotmail.com”  
Esperado: Deve exibir mensagem de “paciente não encontrado”  
Resultado: Exibição de mensagem

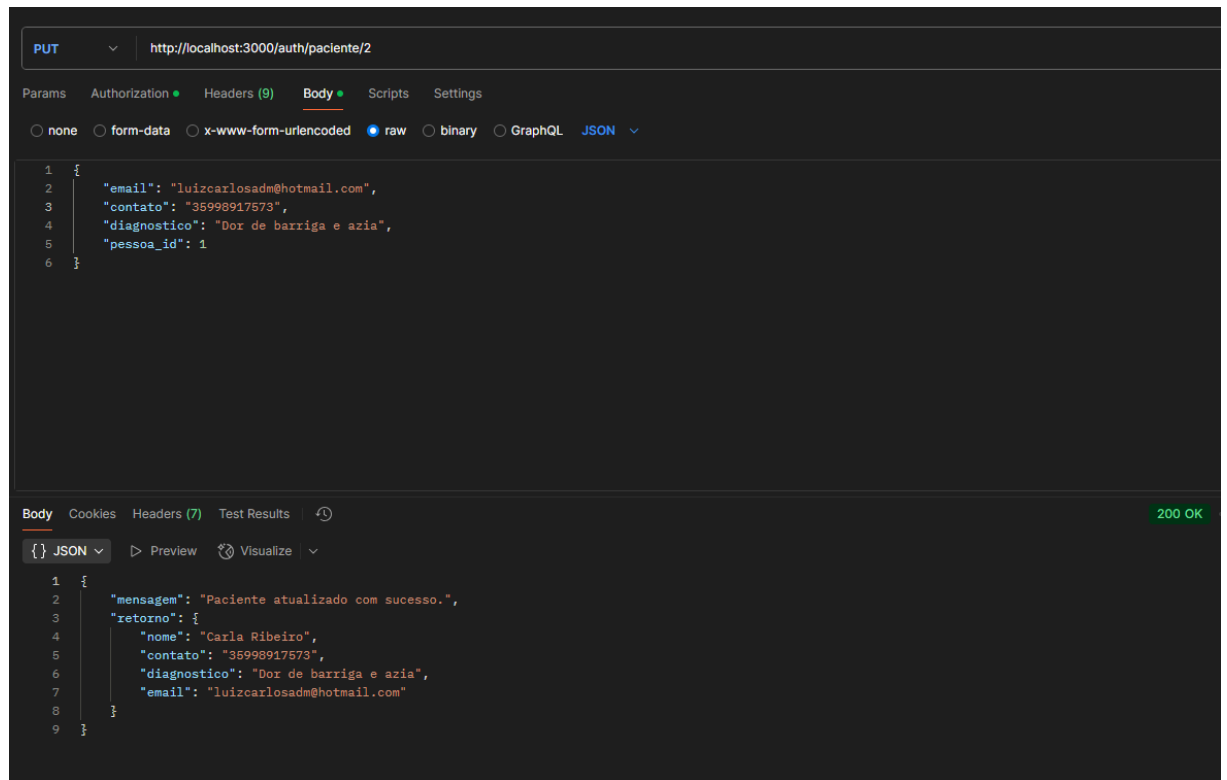


b) Quando encontra o paciente e atualiza os dados, deve retornar as informações atualizadas

Entrada: Dados a serem atualizados como, email, contato, diagnóstico e o id da pessoa.

Esperado: Deve exibir os dados do paciente atualizado com mensagem de sucesso.

Resultado: Exibição de mensagem e dados atualizados



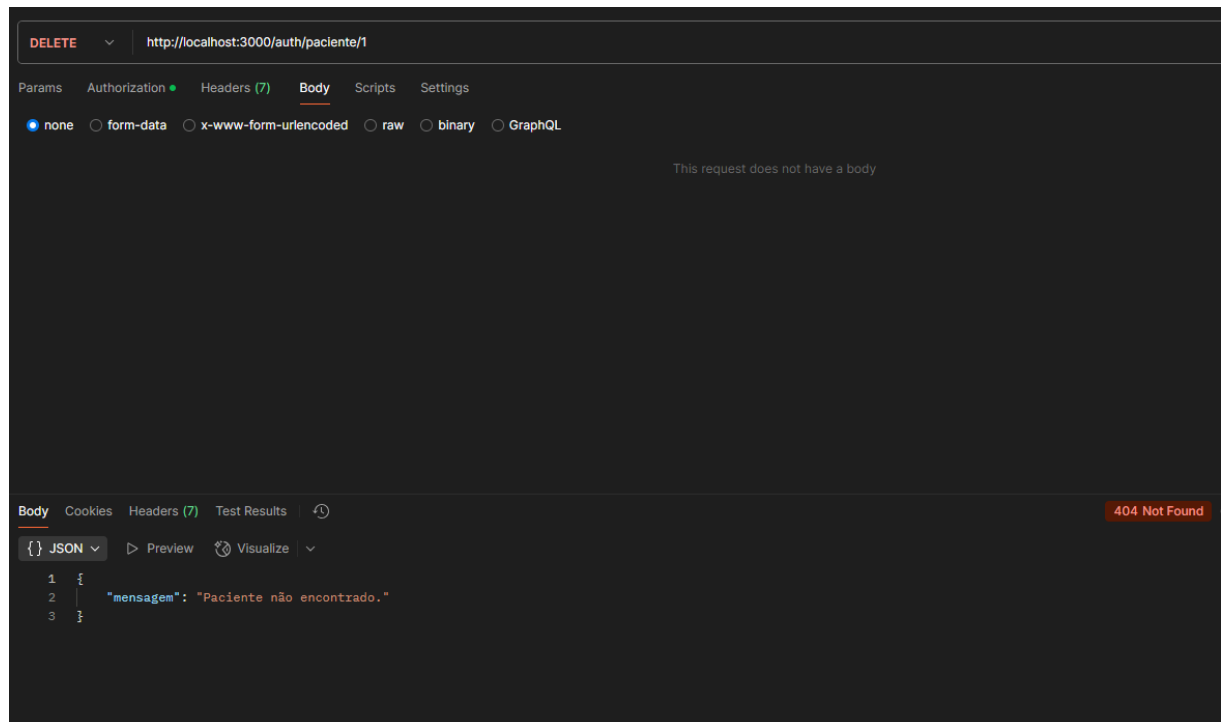
## CT011 Excluir paciente

- a) Exibe mensagem quando não encontra um paciente

Entrada: nenhum dado necessário

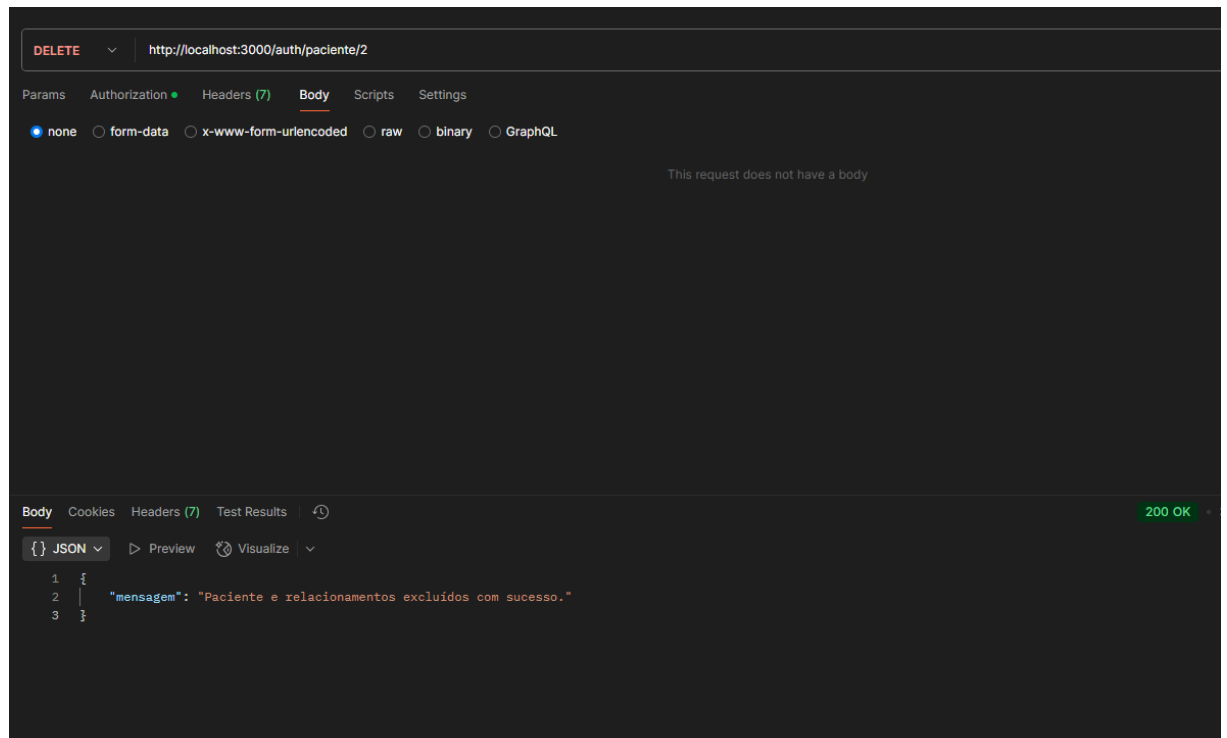
Esperado: Deve exibir mensagem de “paciente não encontrado”

Resultado: Exibição de mensagem



- b) Exibe mensagem de sucesso ao excluir paciente e seus relacionamentos no banco
- Entrada: nenhum dado necessário
- Esperado: Deve exibir mensagem de “paciente e relacionamentos excluídos com sucesso”
- Resultado: Exibição de mensagem





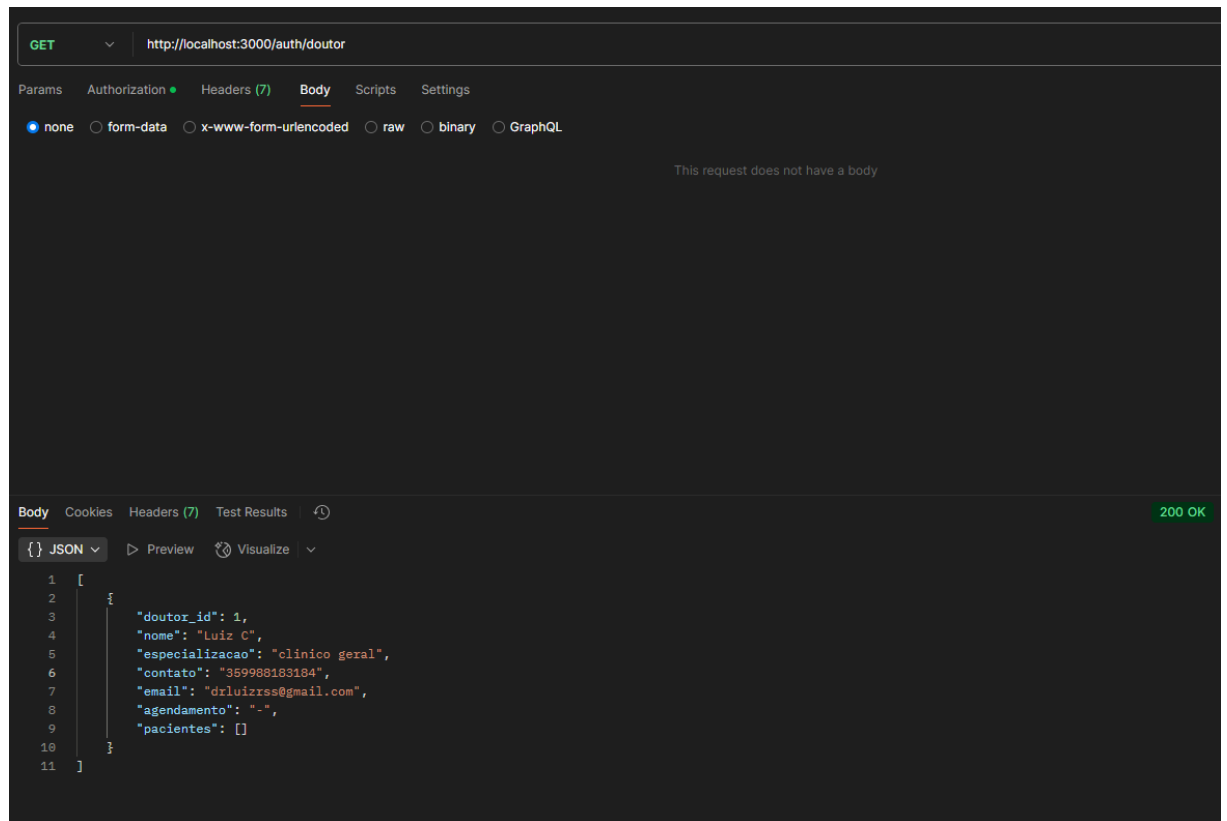
## CT 011 Listar doutores

- a) O sistema deve exibir todos doutores cadastrados no sistema

Entrada: nenhum dado necessário

Esperado: Deve exibir a lista de doutores com as informações de doutor ,nome, especialização, contato e-mail, agendamento e pacientes.

Resultado: Exibir lista de doutores



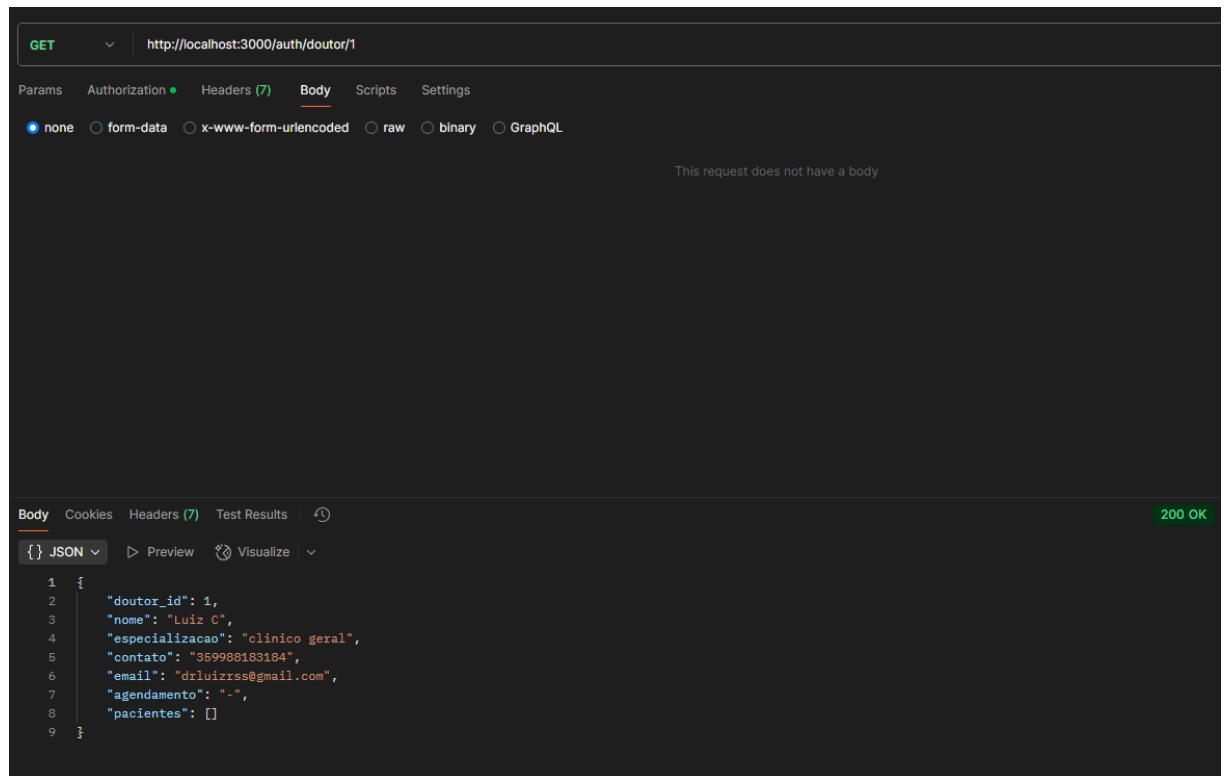
## CT012 Obter doutor por id

- a) O sistema deve exibir o doutor encontrado

Entrada: nenhum dado necessário

Esperado: Deve exibir os dados do doutor como nome, especialização, contato e-mail, agendamento e pacientes.

Resultado: Exibir informações do doutor pesquisado

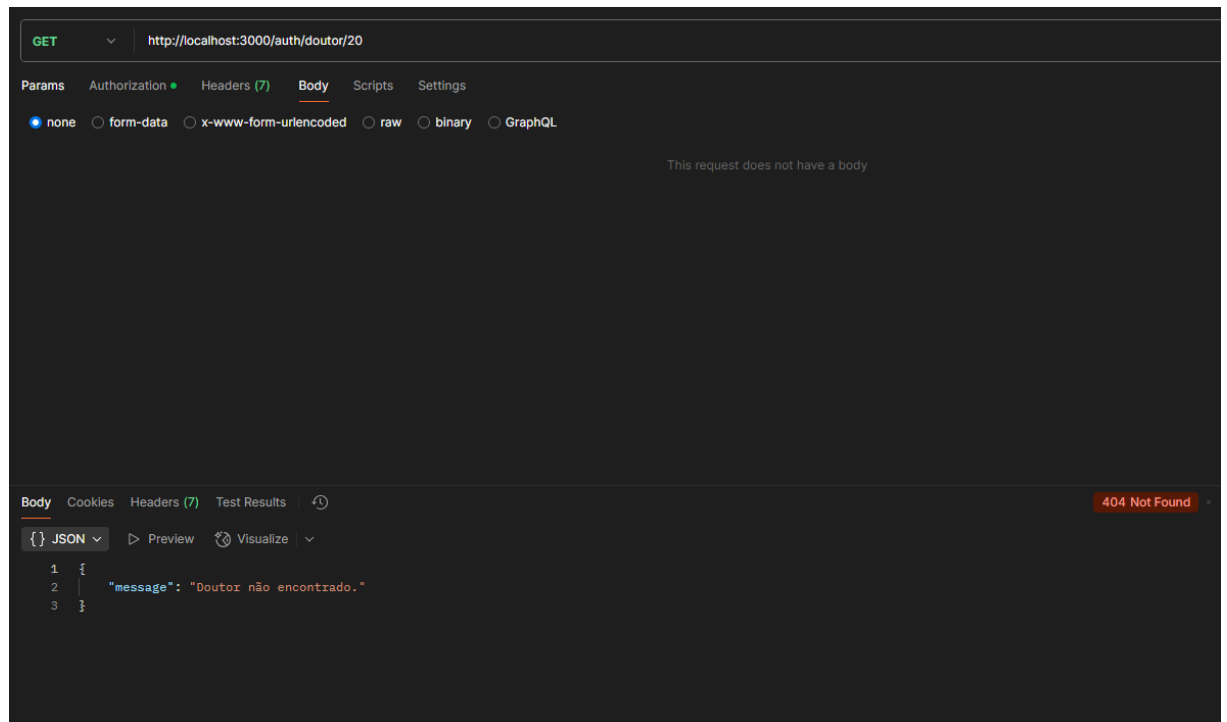


b) O sistema deve exibir uma mensagem caso não encontre um doutor com aquele id

Entrada: nenhum dado necessário

Esperado: Deve exibir mensagem de “doutor não encontrado”

Resultado: Exibição de mensagem



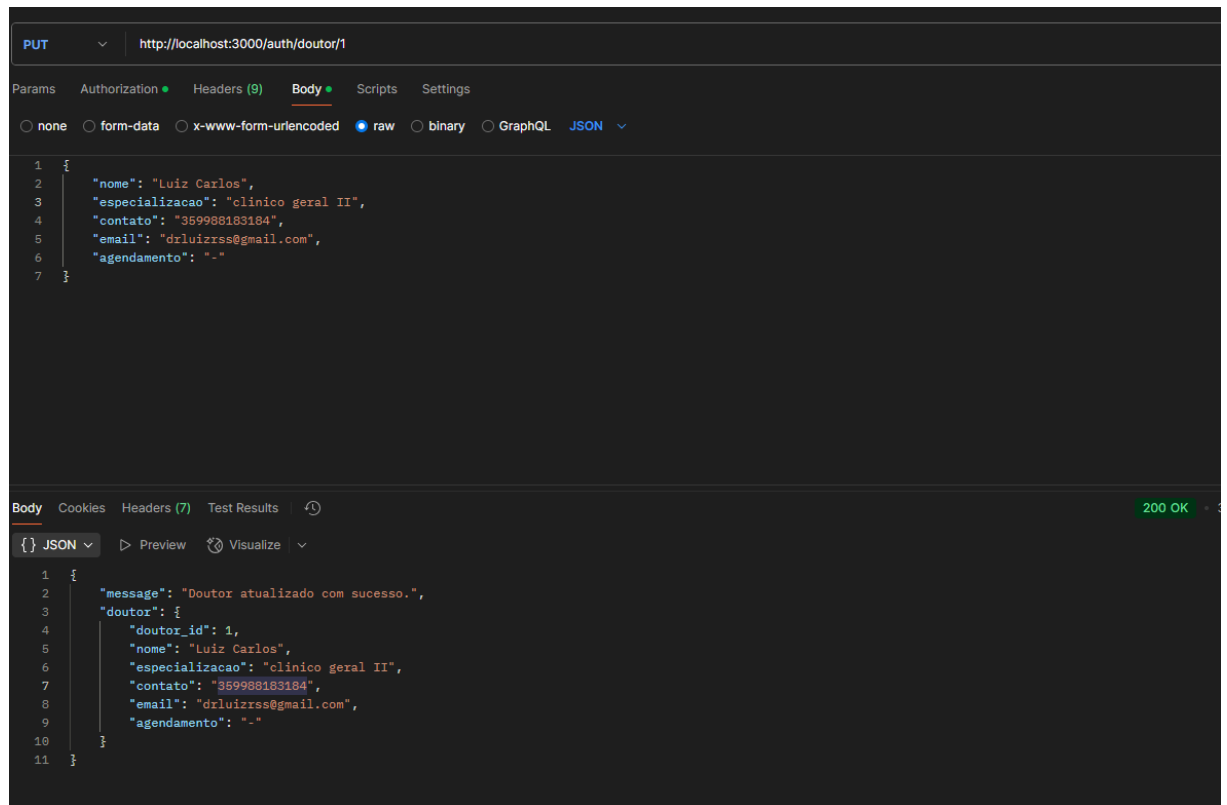
### CT013 Editar doutor

- a) Caso atualize com sucesso o sistema deve retornar os dados atualizados do doutor

Entrada: nome, especialização, contato, e-mail, agendamento.

Esperado: Deve exibir mensagem de “doutor atualizado com sucesso” mostrando nome, especialização, contato, e-mail e agendamento.

Resultado: Exibição dos dados atualizados

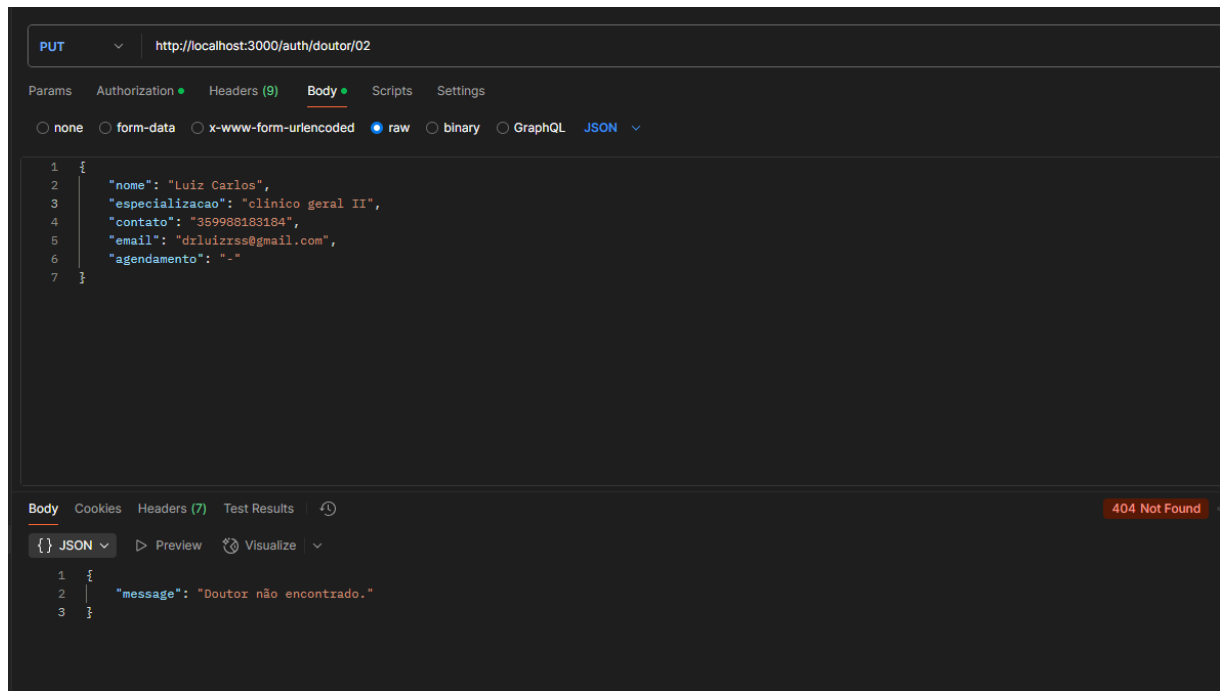


b) Caso não encontre o doutor para atualizar, o sistema deve exibir uma mensagem informando

Entrada: nome, especialização, contato, e-mail, agendamento.

Esperado: Deve exibir mensagem de “doutor não encontrado”

Resultado: Exibição de mensagem



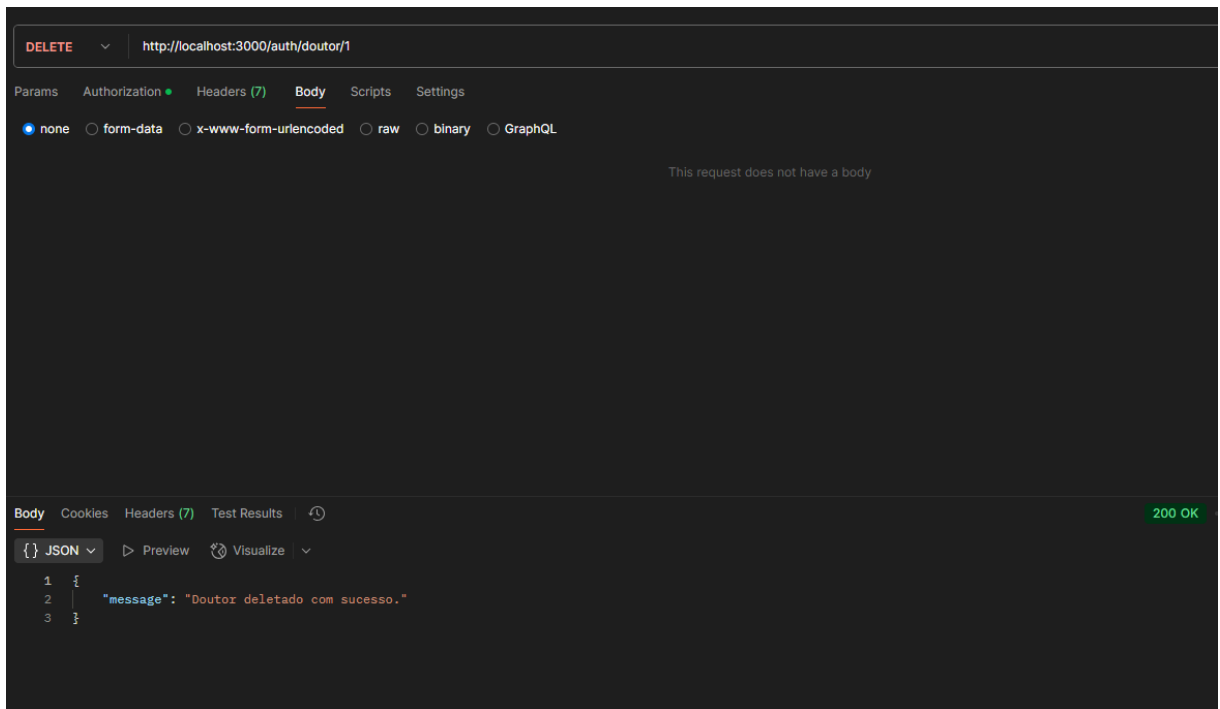
#### CT014 Excluir doutor

- a) O sistema deve exibir uma mensagem caso excluído com sucesso

Entrada: nenhum dado necessário

Esperado: Deve exibir mensagem “doutor deletado com sucesso”

Resultado: Exibição de mensagem

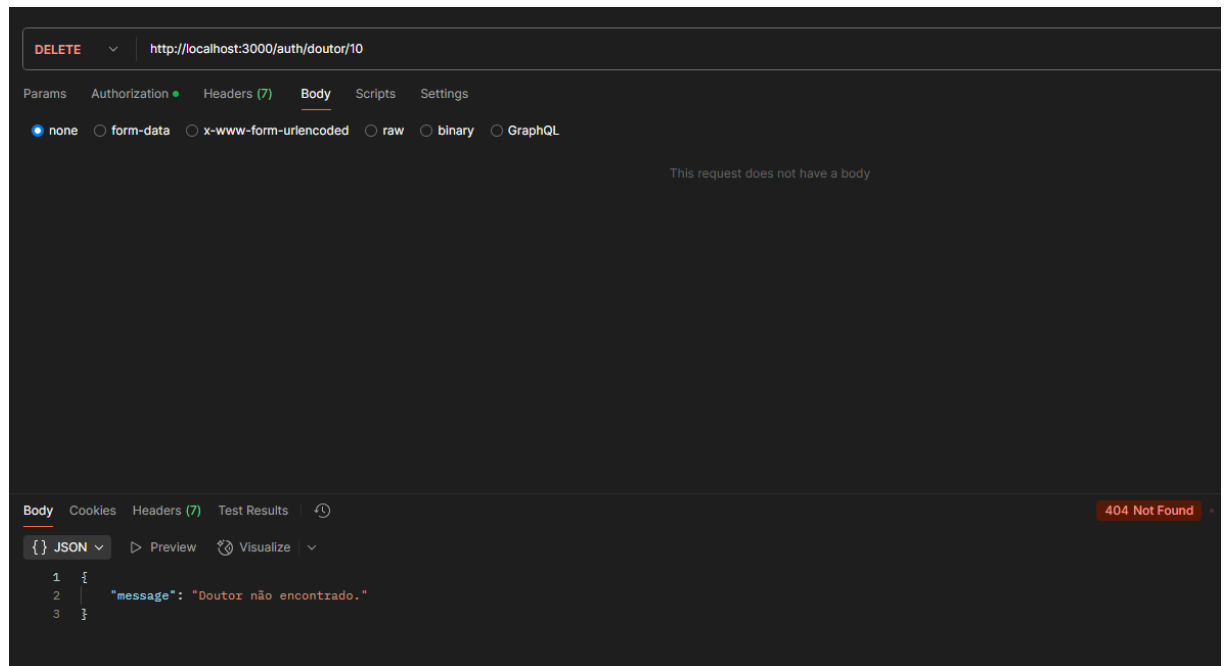


b) Caso não encontre o doutor, deve exibir uma mensagem

Entrada: nenhum dado necessário

Esperado: Deve exibir mensagem “doutor não encontrado”

Resultado: Exibição de mensagem



#### CT015 Adicionar paciente para doutor

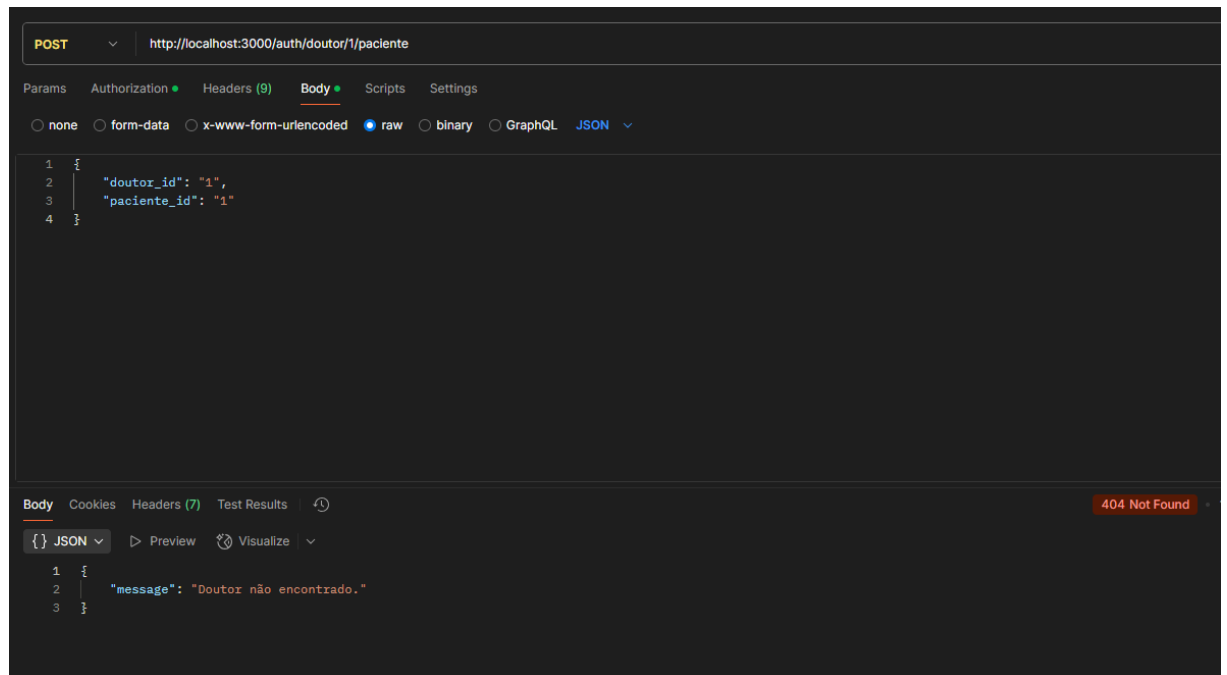
- a) Para associar deve informar o id do paciente e o doutor a serem associados. Caso não encontre, exiba uma mensagem.

Entrada: nome doutor e paciente

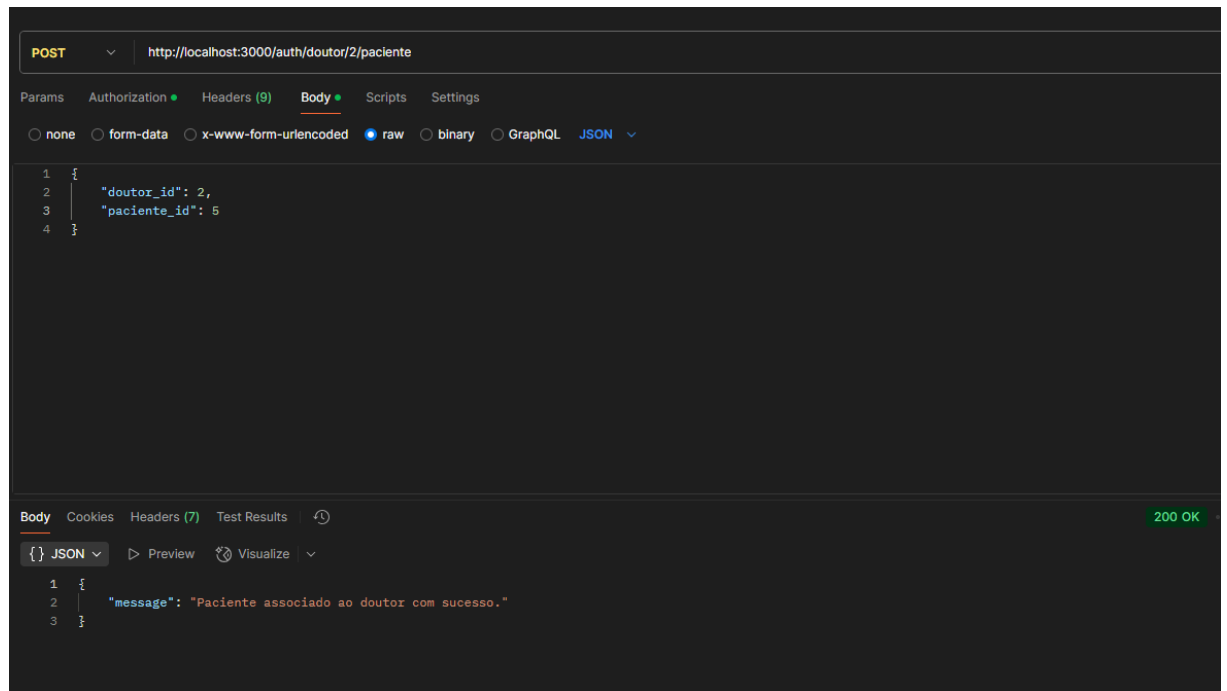
Esperado: Deve exibir mensagem “doutor não encontrado”

Resultado: Exibição de mensagem





- b) Em caso de sucesso, deve exibir uma mensagem informado.  
Os dados de entrada são doutor id e paciente id  
Entrada: nome, doutor e paciente  
Esperado: Deve exibir mensagem “paciente associado ao doutor com sucesso”.
- c) Resultado: Exibição de mensagem



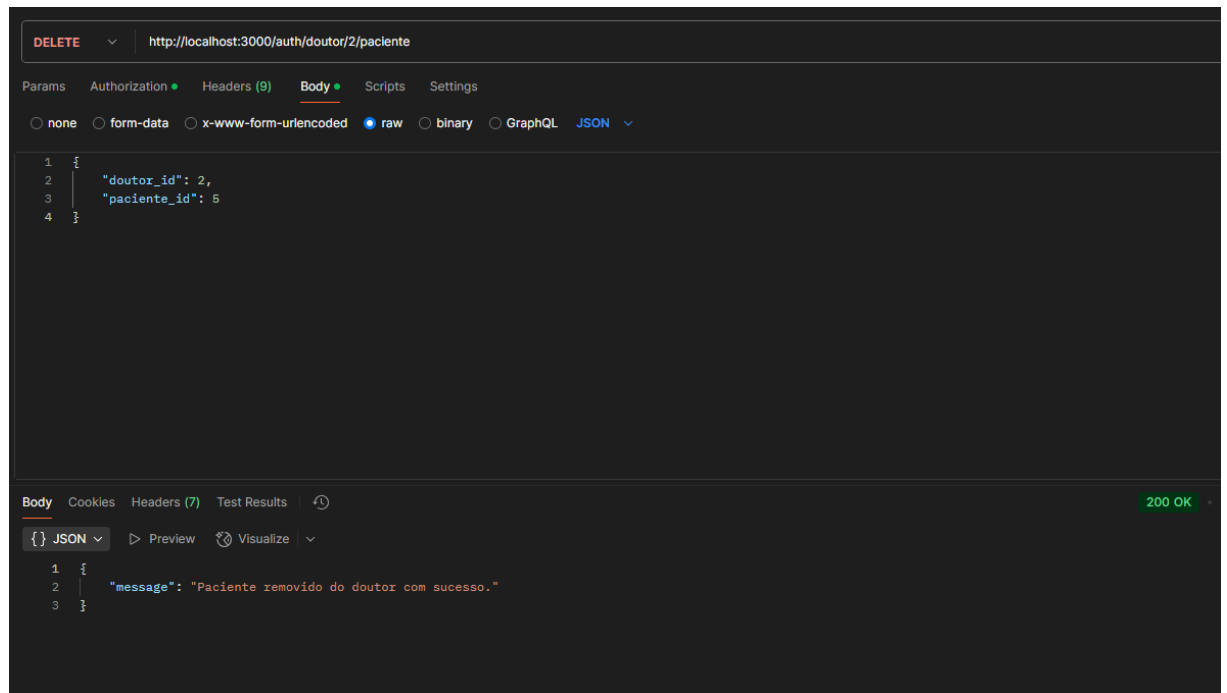
#### CT016 Excluir paciente do doutor

- a) Deve ser fornecido doutor id e paciente id como dados de entrada. O resultado esperado é uma mensagem informando se foi desmarcado a associação com sucesso.

Entrada: nome doutor e paciente

Esperado: Deve exibir mensagem “paciente removido do doutor com sucesso”

Resultado: Exibição de mensagem

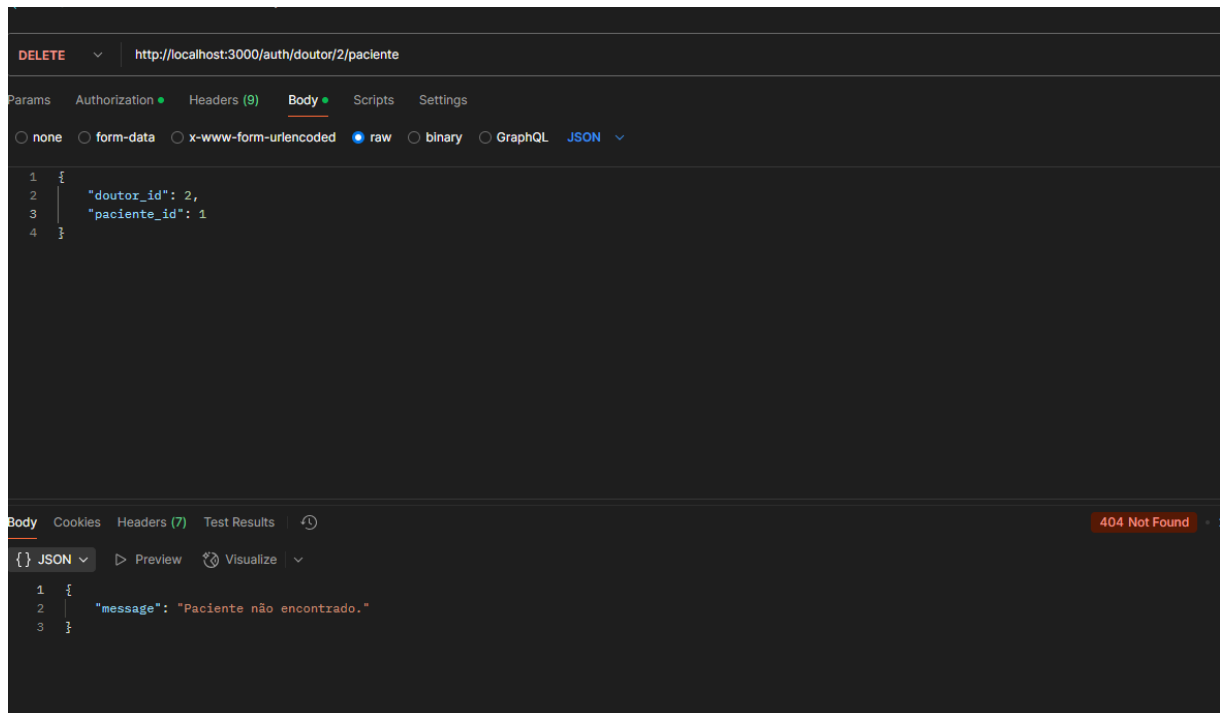


- b) Caso não exista um paciente ou um doutor com esse id, o sistema retorna também uma mensagem informando.

Entrada: nome, doutor e paciente

Esperado: Deve exibir mensagem “paciente não encontrado”

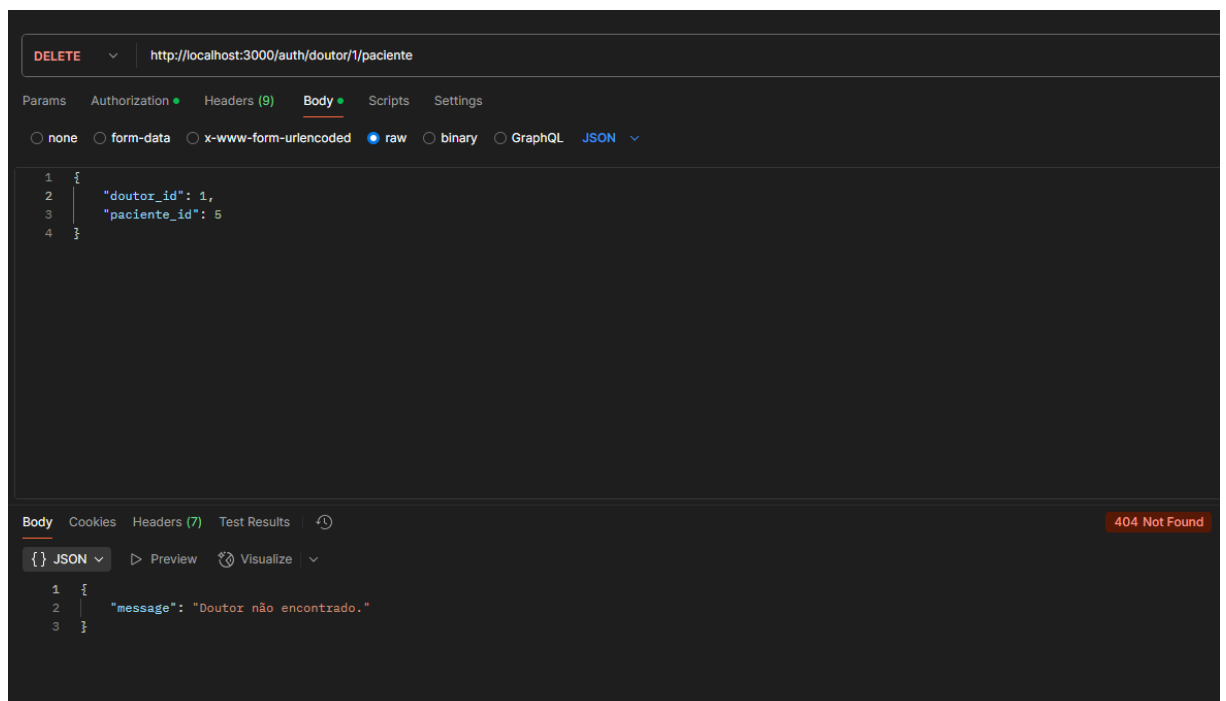
Resultado: Exibição de mensagem



Entrada: nome, doutor e paciente

Esperado: Deve exibir mensagem “doutor não encontrado”

Resultado: Exibição de mensagem



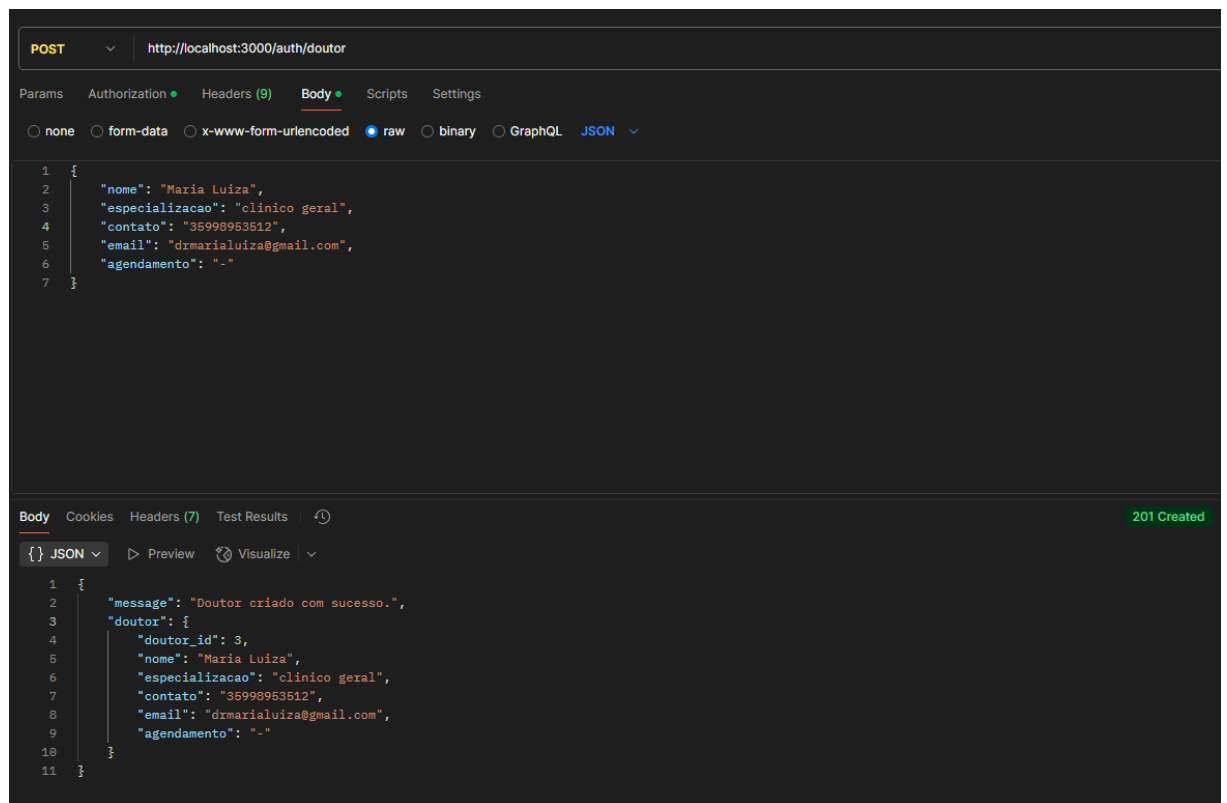
## CT017 Criar doutor

- a) O sistema deve retornar os dados do doutor criado quando houver sucesso na criação

Entrada: nome, especialização, contato, e-mail, agendamento.

Esperado: Deve exibir mensagem “doutor criado com sucesso” mostrando nome, especialização, contato, e-mail e agendamento.

Resultado: Exibição de mensagem e os dados do doutor criado.

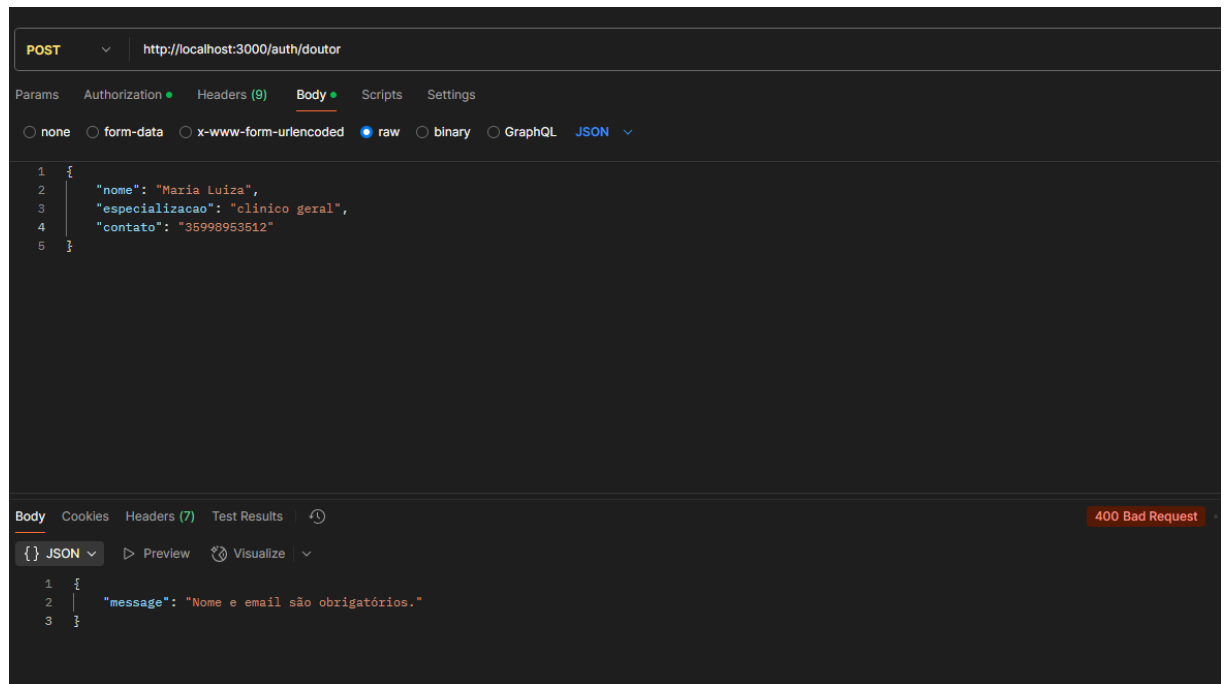


- b) Em caso de falha, o sistema deve retornar uma mensagem informando o motivo

Entrada: nome, especialização, contato

Esperado: Deve exibir mensagem “nome e-mail são obrigatórios”

Resultado: Exibição de mensagem

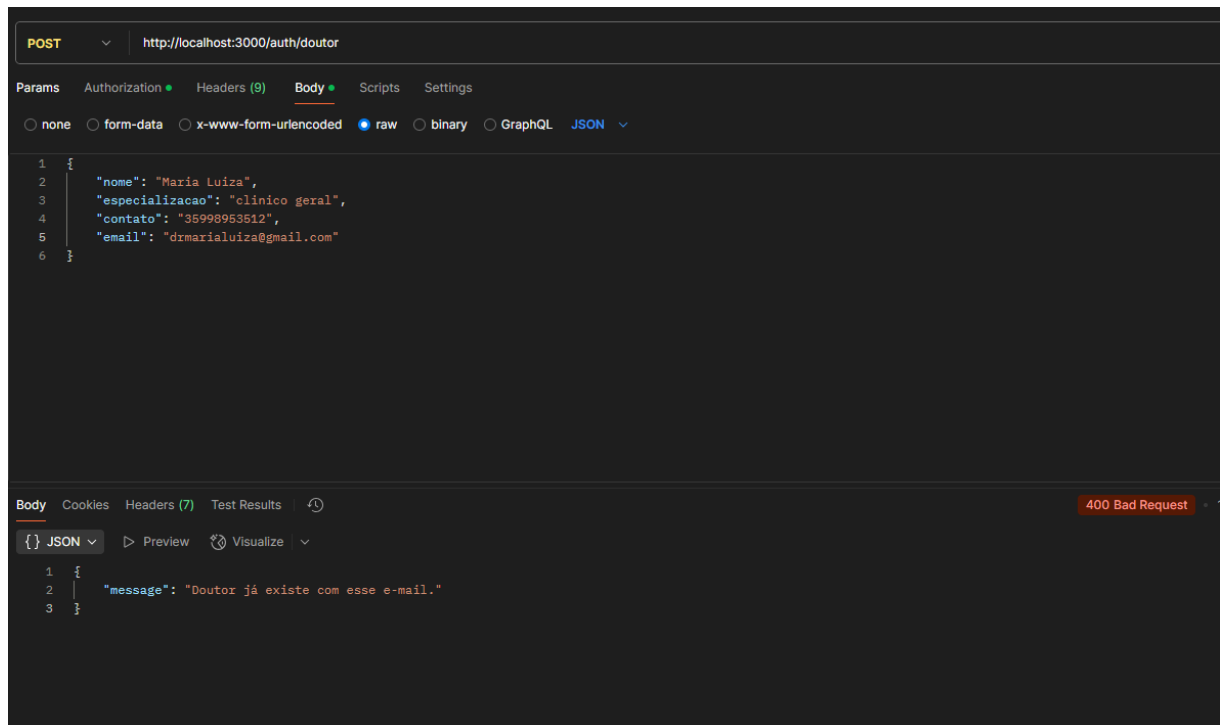


c) Em caso de email já cadastrado, deve retornar uma mensagem informando

Entrada: nome, especialização, contato

Esperado: Deve exibir mensagem “doutor já existe com esse e-mail”

Resultado: Exibição de mensagem



## Conclusão

A realização deste projeto representou um grande desafio, especialmente por eu ainda não possuir experiência na área. Cada etapa exigiu esforço e dedicação, e em diversos momentos foi necessário buscar ajuda externa para superar dificuldades, principalmente durante a criação dos primeiros CRUDs, que apresentaram diversos erros.

Apesar das dificuldades iniciais, à medida que avancei no desenvolvimento, os processos se tornaram mais compreensíveis, especialmente após a finalização do primeiro CRUD. O aspecto mais complexo foi lidar com os relacionamentos entre tabelas no banco de dados MySQL, o que exigiu estudo e persistência para encontrar soluções eficazes.

No entanto, todo o esforço resultou em um aprendizado significativo, tanto no uso do Node.js quanto na modelagem e desenvolvimento de bancos de dados relacionais. Este projeto contribuiu de forma relevante para minha formação e ampliou minha confiança na área de desenvolvimento.

#### Referências

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *UML – Guia do Usuário*. 2. ed. Porto Alegre: Bookman, 2005.

SOMMERVILLE, Ian. *Engenharia de Software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011.









