

# CHALLENGE 2024

Plusoft - Mastering Relational and Non-Relational Database

## **Integrantes:**

**RM551261 - Giovanni Sguizzardi**

**RM98057 - Nicolas E. Inohue**

**RM99841 - Marcel Prado Soddano**

**RM552302 - Samara Moreira**

**RM552293 - Vinicius Monteiro**

# 1. Documentação do Projeto e Justificativa de uso do MongoDB

## 1.1. Introdução ao Projeto

O sistema de gestão de saúde que estamos desenvolvendo é voltado para facilitar o gerenciamento e organização dos dados de pacientes, médicos, exames, consultas, transações financeiras, procedimentos e outras informações fundamentais para uma instituição de saúde. Ele permitirá o controle eficiente e seguro de diferentes entidades relacionadas à saúde, como prescrições, convênios e postos de saúde.

Este sistema visa fornecer uma solução escalável e de alto desempenho, com consultas rápidas e estruturação inteligente das informações. O objetivo é garantir uma gestão integrada, permitindo que profissionais de saúde tenham acesso rápido a dados essenciais para a tomada de decisão, ao mesmo tempo em que se oferece segurança e confiabilidade para os pacientes.

## 1.2. Justificativa para Escolha do MongoDB

A escolha do **MongoDB** como banco de dados NoSQL para este projeto se baseia em suas diversas vantagens em comparação aos bancos de dados relacionais. Abaixo, destacamos as justificativas mais importantes para essa escolha:

### 1. Flexibilidade no Modelo de Dados:

- O MongoDB utiliza documentos no formato JSON/BSON, permitindo que os dados sejam armazenados de forma não estruturada ou semiestruturada. Isso é essencial para lidar com informações complexas, como registros médicos que podem variar de paciente para paciente.

- Estruturas de dados aninhadas e arrays facilitam o armazenamento de dados relacionados, como o histórico de consultas de um paciente ou a lista de medicamentos prescritos em uma receita.

### 2. Alto Desempenho e Escalabilidade:

- O MongoDB oferece suporte a **índices** eficientes, permitindo que consultas em atributos críticos (como CPF de um paciente ou especialidade de um médico) sejam rápidas e escaláveis.

- Com o **sharding** (particionamento horizontal), é possível distribuir os dados em diferentes servidores, garantindo que o sistema continue rápido mesmo com um volume crescente de informações.

### 3. Agilidade no Desenvolvimento:

- A estrutura flexível do MongoDB acelera o desenvolvimento, permitindo que a equipe adicione novos atributos aos documentos sem a necessidade de alterar a estrutura do banco, o que seria complexo em um banco relacional.

- O uso de bibliotecas como **Pymongo** no Python simplifica as operações CRUD e facilita a integração com sistemas externos.

### 4. Alta Disponibilidade e Recuperação de Desastres:

- Com **Replica Sets**, o MongoDB permite a replicação automática de dados entre servidores. Assim, em caso de falha de um nó, outro nó pode assumir sem perda de dados.

### 5. Adequação ao Domínio do Projeto:

- No contexto de gestão de saúde, os dados podem ser heterogêneos e mudam com frequência (ex.: novas especialidades, procedimentos e tipos de exames). O MongoDB permite lidar com essa variação de forma mais eficiente do que bancos de dados relacionais tradicionais.

- O sistema pode armazenar registros detalhados, como históricos de prescrições, sem a necessidade de relações complexas entre várias tabelas.

### 6. Integração com Outros Sistemas:

- MongoDB se integra facilmente com APIs RESTful e plataformas como **Node.js**, **Java** e **Python**, possibilitando a criação de uma interface web intuitiva para os usuários.

## 1.3. Objetivos do Projeto

- Armazenar e gerenciar informações de pacientes, médicos, consultas, exames, transações financeiras e prescrições de maneira eficiente.

- Fornecer uma interface de consulta rápida e precisa para médicos e administradores.

- Implementar operações CRUD completas para que os dados possam ser criados, atualizados e excluídos conforme necessário.

- Garantir a segurança e integridade das informações armazenadas.
- Suportar a integração com outros sistemas de saúde através de APIs.

## 1.4. Conclusão

O MongoDB é a escolha ideal para o projeto de gestão de saúde devido à sua flexibilidade, capacidade de lidar com grandes volumes de dados heterogêneos e facilidade de integração com outras plataformas. Este projeto aproveitará todas essas características para garantir que o sistema ofereça alto desempenho, escalabilidade e segurança.

# 2. Modelo de Dados e Justificativas

A seguir, apresento o **modelo de dados** para as coleções principais do sistema de gestão de saúde, com **exemplos JSON de cada coleção** e a justificativa para a estrutura de cada uma. Esses modelos foram projetados para garantir a **eficiência no armazenamento** e a **facilidade de consulta**, considerando a natureza heterogênea das informações e os diferentes relacionamentos entre entidades do sistema.

## 2.1. Estrutura de Documentos: Pacientes

- **Justificativa:** Um paciente pode ter diversos contatos e histórico de consultas e prescrições. A estrutura armazena dados pessoais e de contato, além de permitir o registro de múltiplos convênios e consultas associadas.

**JSON:**

```
{
  "_id": {
    "oid": "6726abcbdb55df79fc7677b"
  },
  "nome": "João da Silva Junior",
  "cpf": "12345678900",
  "data_nascimento": "1980-05-22T00:00:00.000Z",
  "sexo": "M",
  "endereco": {
    "rua": "Rua das Flores",
    "numero": 123,
    "bairro": "Jardim Paulista",
    "cidade": "São Paulo",
    "estado": "SP",
    "pais": "Brasil"
  },
  "contatos": [
    {
      "tipo": "email",
      "valor": "joao@email.com"
    },
    {
      "tipo": "telefone",
      "valor": "(11) 98765-4321"
    }
  ],
  "convênios": [
    "Unimed",
    "Bradesco Saúde"
  ],
  "historico_consultas": [
    {
      "consulta_id": "consult_id_1",
      "data": "2024-10-30T00:00:00.000Z",
      "medico_id": "med_456"
    },
    {
      "consulta_id": "consult_id_2",
      "data": "2024-11-12T00:00:00.000Z",
      "medico_id": "med_123"
    }
  ]
}
```

## 2.2. Estrutura de Documentos: Medicos

- **Justificativa:** Os médicos possuem uma ou mais especialidades e atendem em múltiplos postos de saúde. Também registramos o CRM e outras certificações para garantir a autenticidade.

**JSON:**

```
{
  "_id": {
    "id": "6726ac38bbd55df79fc7677f"
  },
  "nome": "Dra. Ana Souza",
  "crm": "CRM-SP-98765",
  "especialidades": [
    "Cardiologia",
    "Clínica Geral"
  ],
  "postos_saude": [
    {
      "id": "6726ad1ebbd55df79fc76794"
    },
    {
      "id": "6726ad2dbbd55df79fc76795"
    }
  ],
  "contatos": [
    {
      "tipo": "email",
      "valor": "ana.souza@hospital.com"
    },
    {
      "tipo": "telefone",
      "valor": "(11) 99876-5432"
    }
  ],
  "endereco": {
    "rua": "Rua São Paulo",
    "numero": 200,
    "bairro": "Jardins",
    "cidade": "São Paulo",
    "estado": "SP",
    "pais": "Brasil"
  },
  "horarios_atendimento": [
    {
      "dia": "segunda-feira",
      "horario": "08:00 - 12:00"
    },
    {
      "dia": "quarta-feira",
      "horario": "13:00 - 17:00"
    }
  ],
  "status": "Ativo"
}
```

## 2.3. Estrutura de Documentos: Consultas

- **Justificativa:** Armazena detalhes de cada consulta, associando pacientes, médicos, procedimentos e exames.

**JSON:**

```
{
  "_id": {
    "id": "6726ac65bbd55df79fc76782"
  },
  "paciente_id": {
    "id": "6726abcbdbd55df79fc7677b"
  },
  "medico_id": {
    "id": "6726ac38bbd55df79fc7677f"
  },
  "data": "2024-10-30",
  "horario": "14:00",
  "procedimentos": [
    {
      "id": "6726ad63bbd55df79fc7679b"
    },
    {
      "id": "6726ad78bbd55df79fc7679c"
    }
  ],
  "exames": [
    {
      "id": "6726ac98bbd55df79fc76785"
    }
  ],
  "motivo": "Consulta de rotina",
  "observacoes": "Paciente relatou dor de cabeça constante.",
  "status": "Concluída"
}
```

## 2.4. Estrutura de Documentos: Exames

- **Justificativa:** Cada exame contém informações sobre o tipo e o resultado.

**JSON:**

```

{
  "_id": {
    "$oid": "6726ac90bbd55df79fc76785"
  },
  "tipo": "Hemograma completo",
  "data_solicitacao": "2024-10-30",
  "data_resultado": "2024-10-30",
  "resultado": {
    "hemoglobina": "13.5",
    "hematocrito": "40%",
    "leucocitos": "5.500/mm³",
    "plaquetas": "200.000/mm³"
  },
  "paciente_id": {
    "$oid": "6726abdcdbd55df79fc7677b"
  },
  "medico_id": {
    "$oid": "6726ac30bbd55df79fc7677f"
  },
  "observacoes": "Resultados dentro da normalidade",
  "status": "Concluído"
}

```

## 2.5. Estrutura de Documentos: Farmacias

- **Justificativa:** As farmácias são registradas com informações como endereço, contato, serviços para facilitar a busca por prescrições.

**JSON:**

```

{
  "_id": {
    "$oid": "6726accbbd55df79fc7678c"
  },
  "nome": "Farmácia Popular",
  "endereco": {
    "rua": "Rua Central",
    "numero": 100,
    "bairro": "Centro",
    "cidade": "São Paulo",
    "estado": "SP",
    "pais": "Brasil"
  },
  "contatos": [
    {
      "tipo": "email",
      "valor": "farmaciapopular@gmail.com"
    },
    {
      "tipo": "telefone",
      "valor": "(11) 91111-2222"
    }
  ],
  "horario_funcionamento": {
    "segunda_sexta": "08:00 - 20:00",
    "sabado": "08:00 - 14:00",
    "domingo": "Fechado"
  },
  "medicamentos": [
    {
      "$oid": "6726acf5bbd55df79fc7678f"
    },
    {
      "$oid": "6726ad06bbd55df79fc76790"
    }
  ],
  "servicos": [
    "Aplicação de vacinas",
    "Aferição de pressão"
  ]
}

```

## 2.6. Estrutura de Documentos: Medicamentos

- **Justificativa:** Os medicamentos possuem informações detalhadas sobre o nome comercial, princípio ativo e farmácias que o disponibilizam.

**JSON:**

```
{
  "_id": {
    "$oid": "6726acf5bbd55df79fc7678f"
  },
  "nome_comercial": "Paracetamol",
  "principio_ativo": "Paracetamol",
  "dosagem": "500mg",
  "forma": "Comprimido",
  "fabricante": "NeoQuímica",
  "preco": 12,
  "estoque": 50,
  "farmacias": [
    {
      "$oid": "6726accbbd55df79fc7678c"
    },
    {
      "$oid": "6726ace0bbd55df79fc7678d"
    }
  ],
  "indicacoes": [
    "Febre",
    "Dor leve a moderada"
  ],
  "contra_indicacoes": [
    "Hipersensibilidade ao paracetamol"
  ]
}
```

## 2.7. Estrutura de Documentos: Prescricao

- **Justificativa:** Cada prescrição associa um médico, um paciente e uma lista de medicamentos recomendados.

**JSON:**

```
{
  "_id": {
    "$oid": "6726ad4fbbd55df79fc76799"
  },
  "paciente_id": {
    "$oid": "6726abcbdd55df79fc7677b"
  },
  "medico_id": {
    "$oid": "6726ac30bbd55df79fc7677f"
  },
  "data": "2024-10-30",
  "medicamentos": [
    {
      "id": {
        "$oid": "6726acf5bbd55df79fc7678f"
      },
      "dosagem": "500mg",
      "frequencia": "8 em 8 horas"
    },
    {
      "id": {
        "$oid": "6726ad66bbd55df79fc76790"
      },
      "dosagem": "10mg",
      "frequencia": "1 vez ao dia"
    }
  ],
  "observacoes": "Manter acompanhamento médico caso sintomas persistam",
  "status": "Em uso"
}
```

## 2.8. Estrutura de Documentos: Procedimentos

- **Justificativa:** Lista os procedimentos realizados durante consultas, associados a pacientes e médicos.

**JSON:**

```
{
  "_id": {
    "$oid": "6726ad63bbd55df79fc7679b"
  },
  "descricao": "Ecocardiograma",
  "paciente_id": {
    "$oid": "6726abcbdd55df79fc7677b"
  },
  "medico_id": {
    "$oid": "6726ac30bbd55df79fc7677f"
  },
  "data": "2024-10-30",
  "local_realizacao": "Laboratorio Lavosier",
  "resultado": "Sem alterações significativas",
  "observacoes": "Nenhuma observação adicional",
  "status": "Concluído"
}
```

## 2.9. Estrutura de Documentos: Postos de Saúde

- **Justificativa:** Postos de saúde são registrados com informações como endereço, contato, serviços e os médicos que atendem neles.

**JSON:**

```
{
  "_id": {
    "$oid": "6726ad1ebbd55df79fc76794"
  },
  "nome": "Posto de Saúde Central",
  "endereco": {
    "rua": "Rua Fausto",
    "numero": 452,
    "bairro": "Moimho Velho",
    "cidade": "São Paulo",
    "estado": "SP",
    "pais": "Brasil"
  },
  "contatos": [
    {
      "tipo": "email",
      "valor": "saudecentral@gmail.com"
    },
    {
      "tipo": "telefone",
      "valor": "(11) 97898-1289"
    }
  ],
  "horario_funcionamento": {
    "segunda_sexta": "08:00 - 18:00",
    "sabado": "08:00 - 12:00",
    "domingo": "Fechado"
  },
  "servicos": [
    "Consulta médica",
    "Vacinação",
    "Exames laboratoriais"
  ],
  "especialidades_disponiveis": [
    "Clínica Geral",
    "Cardiologia",
    "Neurologia"
  ],
  "medicos": [
    {
      "$oid": "6726ac38bbd55df79fc7677f"
    },
    {
      "$oid": "6726ac4dbbd55df79fc76780"
    }
  ]
}
```

## 2.10. Estrutura de Documentos: Transações Financeiras

- **Justificativa:** As transações registram informações de pagamento associadas a consultas, procedimentos e exames.

**JSON:**

```
{
  "_id": {
    "$oid": "6726ad96bbd55df79fc7679f"
  },
  "descricao": "Ecocardiograma",
  "valor": 200,
  "data": "2024-10-30",
  "paciente_id": {
    "$oid": "6726abcbdd55df79fc7677b"
  },
  "procedimento_id": "proc_1",
  "forma_pagamento": "Cartão de Crédito",
  "status": "Pago",
  "detalhes_pagamento": {
    "numero_cartao": "**** * 1234",
    "nome_titular": "João da Silva",
    "data_expiracao": "12/26"
  },
  "nota_fiscal": "NF-123456",
  "observacoes": "Pagamento realizado com sucesso"
}
```

## 2.11. Justificativa Geral do Modelo

- **Modelo Flexível:** O uso de documentos JSON permite armazenar informações variadas, como diferentes tipos de exames e prescrições.









- **Consulta Otimizada:** A estrutura aninhada facilita consultas complexas, por exemplo, buscar todas as consultas de um paciente ou listar os médicos de um posto de saúde.



- **Escalabilidade:** Como o MongoDB permite **sharding**, as coleções podem ser distribuídas por diferentes servidores para manter o desempenho conforme o volume de dados aumenta.

### 3. Construção de Dados, Interface e Operações CRUD

### 3.1. Criação dos 10 Documentos JSON/BSON (Ex: Pacientes):

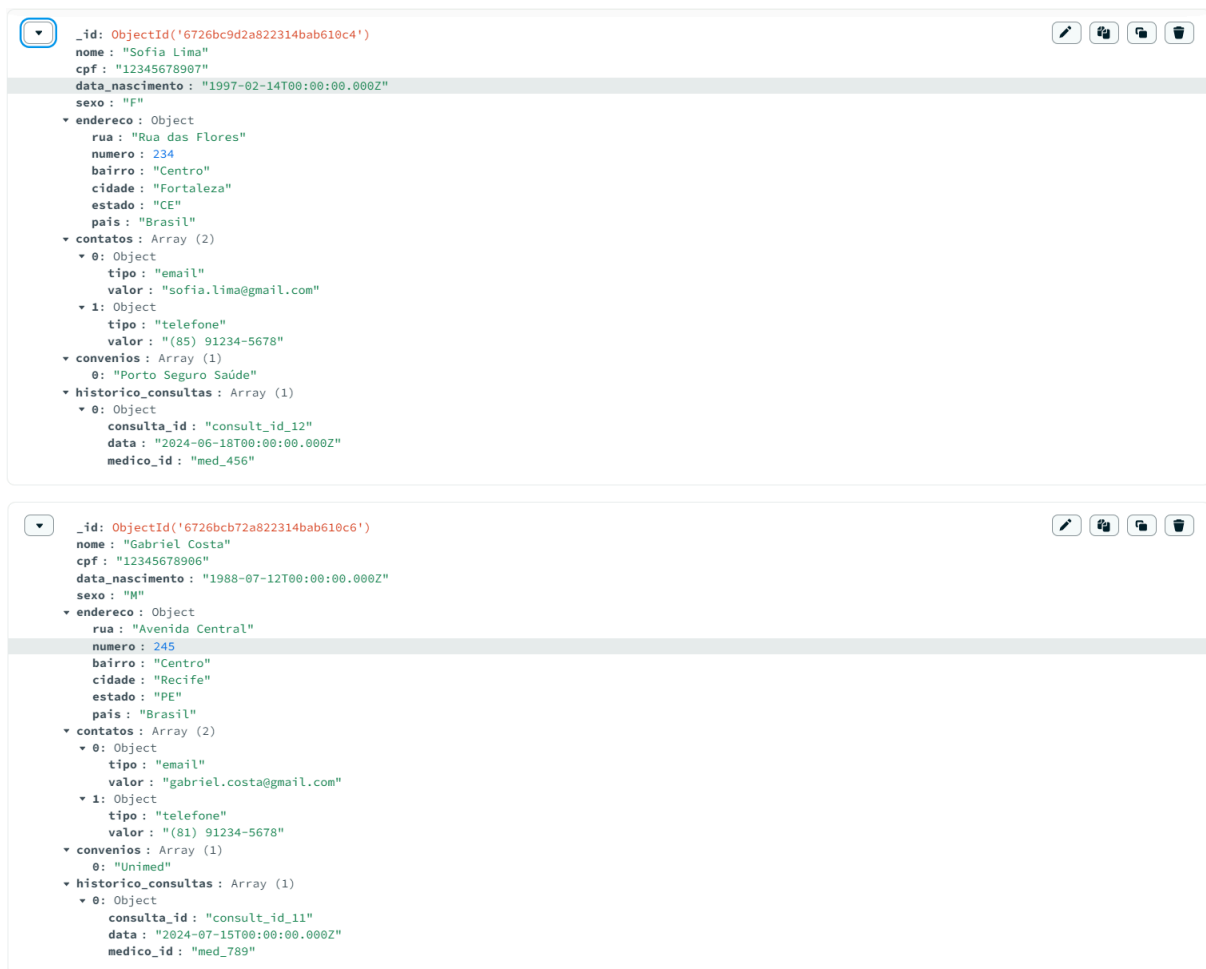
▼	<pre> _id: ObjectId('6726abcbd55df79fc7677b') nome: "João da Silva" cpf: "12345678900" data_nascimento: "1980-05-22T00:00:00.000Z" sexo: "M" ▼ endereco: Object   rua: "Rua das Flores"   numero: 123   bairro: "Jardim Paulista"   cidade: "São Paulo"   estado: "SP"   pais: "Brasil" ▼ contatos: Array (2)   ▼ 0: Object     tipo: "email"     valor: "joao@email.com"   ▼ 1: Object     tipo: "telefone"     valor: "(11) 98765-4321" ▼ convenios: Array (2)   0: "Unimed"   1: "Bradesco Saúde" ▼ historico_consultas: Array (2)   ▼ 0: Object     consulta_id: ObjectId('6726ac5bbd55df79fc76782')     data: "2024-10-30T00:00:00.000Z"     medico_id: ObjectId('6726ac30bbd55df79fc7677f')   ▼ 1: Object     consulta_id: ObjectId('6726ac76bbd55df79fc76783')     data: "2024-11-12T00:00:00.000Z"     medico_id: ObjectId('6726ac40bbd55df79fc76780') </pre>	   
▼	<pre> _id: ObjectId('6726ac8cbdd55df79fc7677d') nome: "Maria Oliveira" cpf: "98765432100" data_nascimento: 1990-03-15T00:00:00.000+00:00 sexo: "F" ▼ endereco: Object   rua: "Avenida das Américas"   numero: 456   bairro: "Centro"   cidade: "Rio de Janeiro"   estado: "RJ"   pais: "Brasil" ▼ contatos: Array (2)   ▼ 0: Object     tipo: "email"     valor: "maria.oliveira@gmail.com"   ▼ 1: Object     tipo: "telefone"     valor: "(21) 91234-5678" ▼ convenios: Array (2)   0: "Amil"   1: "SulAmérica Saúde" ▼ historico_consultas: Array (2)   ▼ 0: Object     consulta_id: ObjectId('6726b201bbd55df79fc767e5')     data: 2024-09-15T00:00:00.000+00:00     medico_id: ObjectId('6726ac40bbd55df79fc76780')   ▼ 1: Object     consulta_id: ObjectId('6726b2f4bbd55df79fc767f2')     data: 2024-10-20T00:00:00.000+00:00     medico_id: ObjectId('6726ac40bbd55df79fc76780') </pre>	   

```
_id: ObjectId('6726abf9bbd55df79fc7677c')
nome: "Maria do Carmo"
cpf: "12445991200"
data_nascimento: 1999-08-25T00:00:00.000+00:00
sexo: "F"
▼ endereco: Object
  rua: "Av Nazare"
  numero: 1232
  bairro: "Ipiranga"
  cidade: "São Paulo"
  estado: "SP"
  país: "Brasil"
▼ contatos: Array (2)
  ▼ 0: Object
    tipo: "email"
    valor: "maria@email.com"
  ▼ 1: Object
    tipo: "telefone"
    valor: "(11) 91235-4231"
▼ convenios: Array (2)
  0: "Amil Saude"
  1: "NotreDame Intermedica"
▼ historico_consultas: Array (1)
  ▼ 0: Object
    consulta_id: ObjectId('6726ac76bbd55df79fc76783')
    data: 2024-10-30T00:00:00.000+00:00
    medico_id: ObjectId('6726ac4dbbd55df79fc76780')
```

```
_id: ObjectId('6726bc232a822314bab610bf')
nome: "Ana Costa"
cpf: "12345678901"
data_nascimento: 1978-11-22T00:00:00.000Z
sexo: "F"
▼ endereco: Object
  rua: "Rua dos Pinheiros"
  numero: 890
  bairro: "Pinheiros"
  cidade: "São Paulo"
  estado: "SP"
  país: "Brasil"
▼ contatos: Array (2)
  ▼ 0: Object
    tipo: "email"
    valor: "ana.costa@gmail.com"
  ▼ 1: Object
    tipo: "telefone"
    valor: "(11) 91234-5678"
▼ convenios: Array (2)
  0: "Unimed"
  1: "Porto Seguro Saúde"
▼ historico_consultas: Array (1)
  ▼ 0: Object
    consulta_id: "consult_id_6"
    data: "2024-05-12T00:00:00.000Z"
    medico_id: "med_456"
```

```
_id: ObjectId('6726bc472a822314bab610c0')
nome: "Carlos Santos"
cpf: "12345678902"
data_nascimento: 1995-09-14T00:00:00.000Z
sexo: "M"
▼ endereco: Object
  rua: "Rua Brasil"
  numero: 987
  bairro: "Centro"
  cidade: "Porto Alegre"
  estado: "RS"
  país: "Brasil"
▼ contatos: Array (2)
  ▼ 0: Object
    tipo: "email"
    valor: "carlos.santos@gmail.com"
  ▼ 1: Object
    tipo: "telefone"
    valor: "(51) 91234-5678"
▼ convenios: Array (1)
  0: "Intermedica"
▼ historico_consultas: Array (1)
  ▼ 0: Object
    consulta_id: "consult_id_7"
    data: "2024-08-30T00:00:00.000Z"
    medico_id: "med_123"
```

<div><div></div><div><div><div></div><div></div><div></div><div></div></div></div></div> <div><div><div><div></div><div></div><div></div><div></div></div></div><div><div><div></div><div></div><div></div><div></div></div></div><div><div><div></div><div></div><div></div><div></div></div></div><div><div><div></div><div></div><div></div><div></div></div></div></div>
--



### 3.2. Interface

### Funcionalidades da interface (MongoDB Compass):

- **Visualização de Documentos:** Uma tela (MongodbCompass) que exibe documentos salvos no MongoDB.
- **Busca:** Permitir que o usuário filtre documentos por atributos específicos (ex.: buscar clientes por nome ou pedidos por produto)
- **Exemplos:**

- Pacientes com convênio ***Amil***, apenas informando ***id***, ***nome*** e ***convênios***:

MongoDB Compass - 2TDSR/AIHack\_Solutions.pacientes

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (1)

Search connections

2TDSR

AIHack\_Solutions

consultas

exames

farmacias

medicamentos

medicos

pacientes

postossaude

prescricao

procedimentos

transacoesfinanceiras

Alura\_Serie

series

admin

config

local

2TDSR > AIHack\_Solutions > pacientes

Documents Aggregations Schema Indexes 1 Validation

Generate query Explain Reset Find Options

Project { \_id:1, nome:1, convenios: 1 }

Sort { field: -1 } or [ ['field', -1] ]

Collation { locale: 'simple' }

Index Hint { field: -1 }

Max Time MS 60000

Skip 0 Limit 0

EXPORT DATA

25 1 - 3 of 3

\_id: ObjectId('6726abf9bbd55df79fc7677c')

nome: "Maria do Carmo"

convenios: Array (2)

\_id: ObjectId('6726ac8cbbd55df79fc7677d')

nome: "Maria Oliveira"

convenios: Array (2)

\_id: ObjectId('6726bc7b2a822314bab610c3')

nome: "Beatriz Rocha"

convenios: Array (1)

- Médicos com status ***Ativo***, apenas informando ***id***, ***nome***, ***crm*** e ***status***:

MongoDB Compass - 2TDSR/AIHack\_Solutions.medicos

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (1)

Search connections

2TDSR

AIHack\_Solutions

consultas

exames

farmacias

medicamentos

medicos

pacientes

postossaude

prescricao

procedimentos

transacoesfinanceiras

Alura\_Serie

series

admin

config

local

2TDSR > AIHack\_Solutions > medicos

Documents 2 Aggregations Schema Indexes 1 Validation

Generate query Explain Reset Find Options

Project { \_id:1, nome:1, crm:1, status:1 }

Sort { field: -1 } or [ ['field', -1] ]

Collation { locale: 'simple' }

Index Hint { field: -1 }

Max Time MS 60000

Skip 0 Limit 0

EXPORT DATA

25 1 - 2 of 2

\_id: ObjectId('6726ac30bbd55df79fc7677f')

nome: "Dra. Ana Souza"

crm: "CRM-SP-98765"

status: "Ativo"

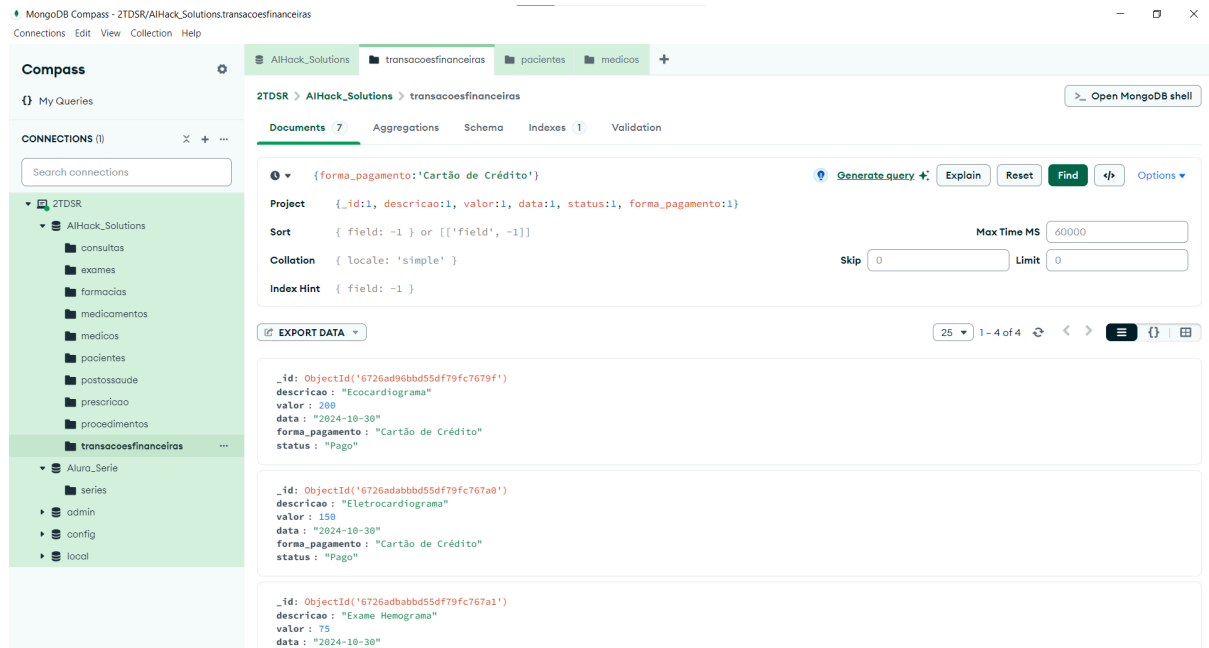
\_id: ObjectId('6726ac4dbbd55df79fc76780')

nome: "Dr. Mario Rodrigues"

crm: "CRM-SP-98867"

status: "Ativo"

- Transações financeiras realizadas com **cartão de crédito**, apenas informando **id**, **descricao**, **valor**, **data**, **status** e **forma de pagamento**:



### 3.3. API

Funcionalidades da API(Python):

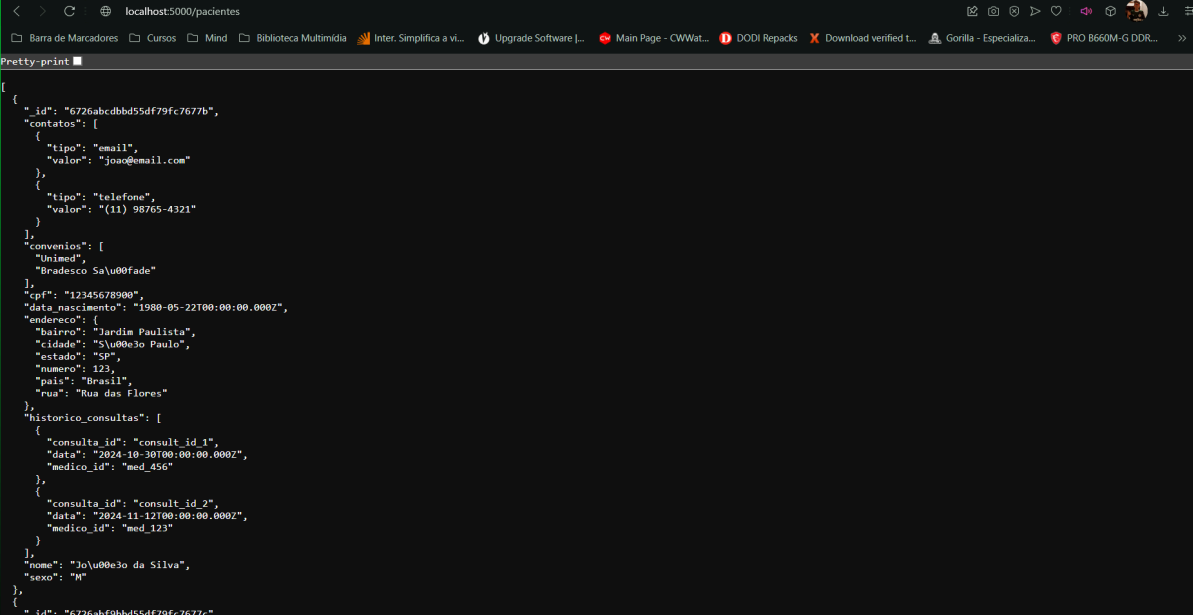
- **CRUD Completo:**
  - **Post:** Adicionar novos documentos.
  - **Get:** Exibe dados da coleção.
  - **Get\_id:** Exibir dados de documentos específicos por ID.
  - **Update:** Atualizar informações em um documento específico.
  - **Delete:** Excluir documentos.
  - **Export:** Exporta as coleções para um arquivo JSON.
  - **Export\_all:** Exporta todas as coleções para um arquivo JSON.

Para dar start no servidor da API, basta abrir a pasta do projeto no terminal e digitar:  
`pip install Flask pymongo`

`python app.py`

A API também pode ser acessada via navegador, digite <http://localhost:5000/colecao> e substitua [colecao](#) pela coleção desejada.

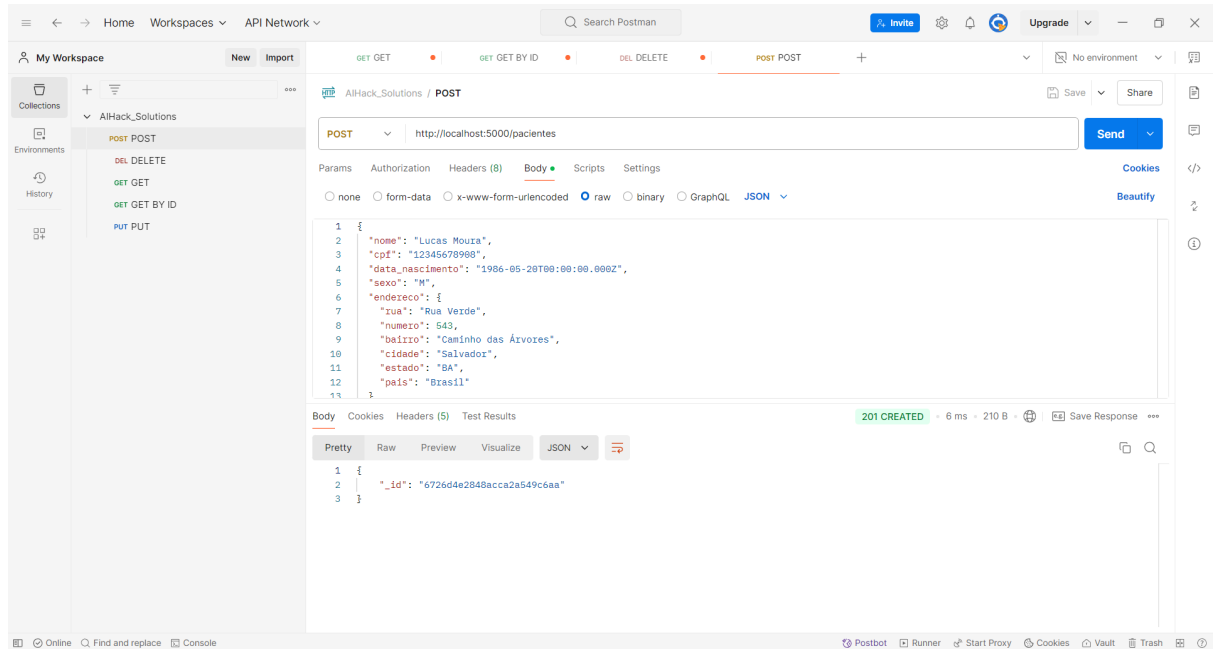
Exemplo de uso no navegador:.



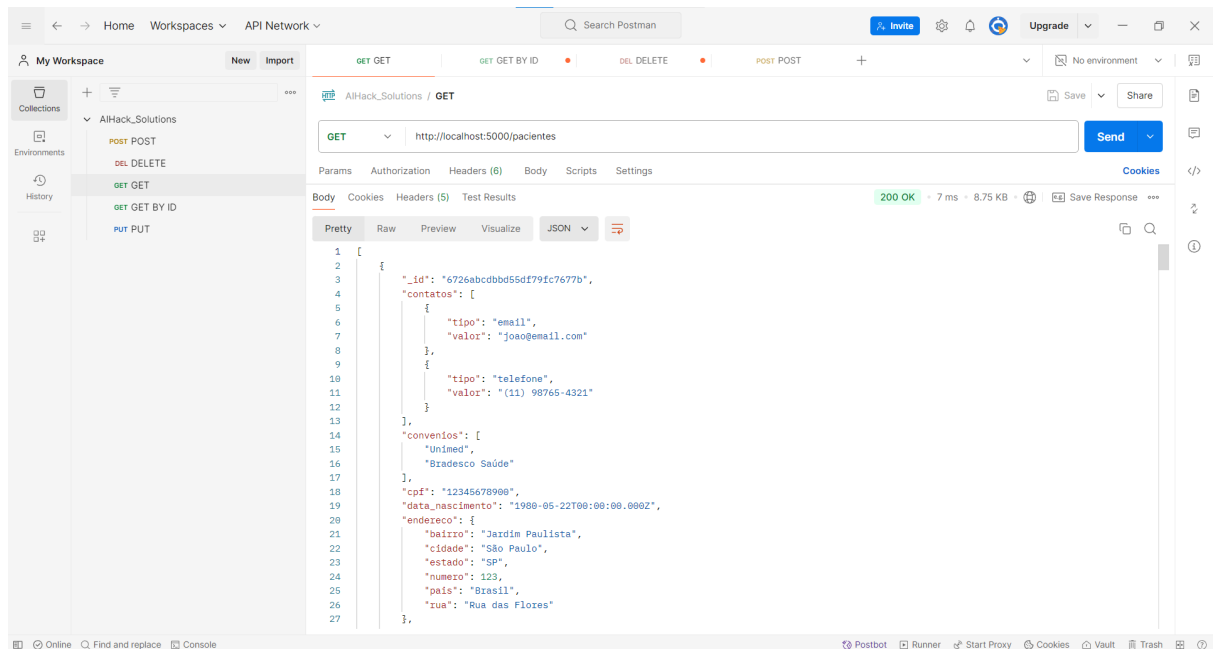
```
[{"_id": "6726abcbdb55df79fc7677b", "contatos": [{"tipo": "email", "valor": "joao@email.com"}, {"tipo": "telefone", "valor": "(11) 98765-4321"}], "convenios": ["Unimed", "Bradesco Sa\u00fade"], "cpf": "12345678900", "data_nascimento": "1980-05-22T00:00:00.000Z", "endereco": {"bairro": "Jardim Paulista", "cidade": "S\u00e3o Paulo", "estado": "SP", "numero": 123, "pais": "Brasil", "rua": "Rua das Flores"}, "historico_consultas": [{"consulta_id": "consult_id_1", "data": "2024-10-30T00:00:00.000Z", "medico_id": "med_456"}, {"consulta_id": "consult_id_2", "data": "2024-11-12T00:00:00.000Z", "medico_id": "med_123"}], "nome": "Jo\u00e3o da Silva", "sexo": "M"}, {"_id": "6726abf9bbd55df79fc7677c", "contatos": []}]
```

- Exemplos demonstrando o funcionamento da **API** para a coleção **Pacientes**:

- **POST**:



- **GET**:





## - GET\_ID:

The screenshot shows a Postman interface with a workspace named "My Workspace". The "Collections" panel on the left shows a collection named "APIHack\_Solutions" with a sub-collection "GET BY ID". The "Environments" panel shows "No environment". The "History" panel shows a list of requests: "POST POST", "DEL DELETE", "GET GET", "GET GET BY ID", and "PUT PUT". The "GET GET BY ID" request is selected, and its details are shown in the main panel. The request is a GET method to the URL "http://localhost:5000/pacientes/6726d4e2848acca2a549c6aa". The response is a 200 OK status, with a response time of 5 ms and a body size of 846 B. The response body is a JSON object representing a patient's data, including contact information, address, and medical history.

```
1 {
2   "_id": "6726d4e2848acca2a549c6aa",
3   "contatos": [
4     {
5       "tipo": "email",
6       "valor": "lucas.moura@gmail.com"
7     },
8     {
9       "tipo": "telefone",
10      "valor": "(71) 92345-6789"
11    }
12  ],
13  "convenios": [
14    {
15      "Cassl"
16    }
17  ],
18  "cpf": "12345678900",
19  "data_nascimento": "1986-05-20T00:00:00.000Z",
20  "endereco": {
21    "bairro": "Caminho das Árvores",
22    "cidade": "Salvador",
23    "estado": "BA",
24    "numero": 543,
25    "pais": "Brasil",
26    "rua": "Rua Verde"
27  },
28  "historico_consultas": [
29    {
30      "data": "2023-10-27T00:00:00.000Z",
31      "descricao": "Consulta de rotina",
32      "medico": "Dr. João da Silva"
33    }
34  ]
35 }
```

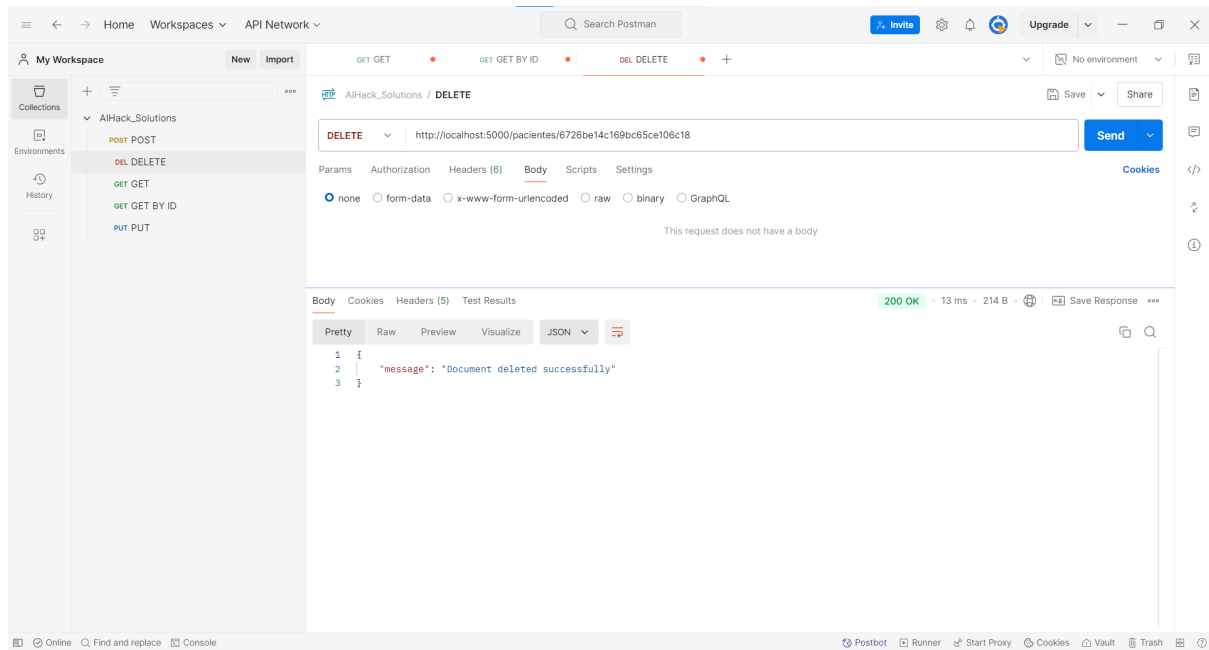
## - PUT:

The screenshot shows a Postman interface with a workspace named "My Workspace". The "Collections" panel on the left shows a collection named "APIHack\_Solutions" with a sub-collection "PUT". The "Environments" panel shows "No environment". The "History" panel shows a list of requests: "POST POST", "DEL DELETE", "GET GET", "GET GET BY ID", and "PUT PUT". The "PUT PUT" request is selected, and its details are shown in the main panel. The request is a PUT method to the URL "http://localhost:5000/pacientes/6726abcd55df79fc7677b". The request body is a JSON object representing a patient's data, including name, CPF, date of birth, sex, address, and medical history. The response is a 200 OK status, with a response time of 6 ms and a body size of 214 B. The response body is a JSON object with a "message" field indicating that the document was updated successfully.

```
1 {
2   "nome": "João da Silva Junior",
3   "cpf": "12345678900",
4   "data_nascimento": "1986-05-22T00:00:00.000Z",
5   "sexo": "M",
6   "endereco": {
7     "rua": "Rua das Flores",
8     "numero": 123,
9     "bairro": "Jardim Paulista",
10    "cidade": "São Paulo",
11    "estado": "SP",
12    "pais": "Brasil"
13  },
14  "historico_consultas": [
15    {
16      "data": "2023-10-27T00:00:00.000Z",
17      "descricao": "Consulta de rotina",
18      "medico": "Dr. João da Silva"
19    }
20  ]
21 }
```

```
1 {
2   "message": "Document updated successfully"
3 }
```

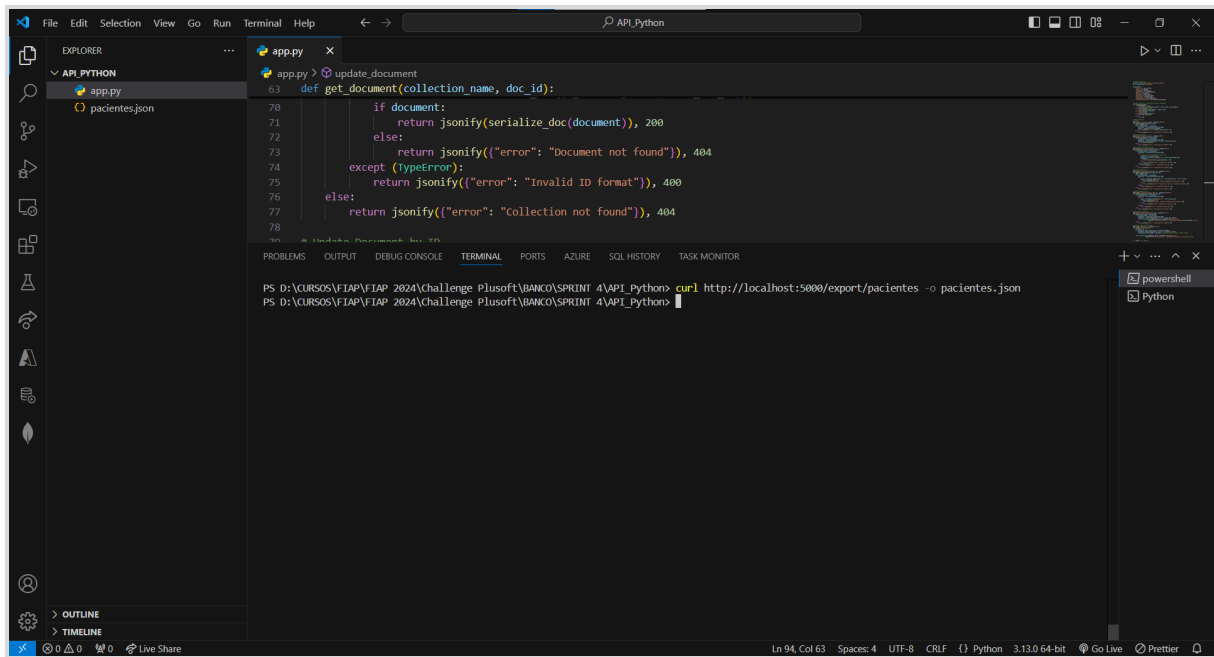
## - DELETE:



## 4. Exportação do Dataset

- É importante alterar **colecão** para o nome da coleção que deseja exportar e alterar **nome\_arquivo** para o nome que você deseja para o arquivo json.
- Exporte um **dataset** dos documentos inseridos no MongoDB utilizando os comandos via terminal:

```
curl http://localhost:5000/export/colecao -o nome_arquivo.json
```

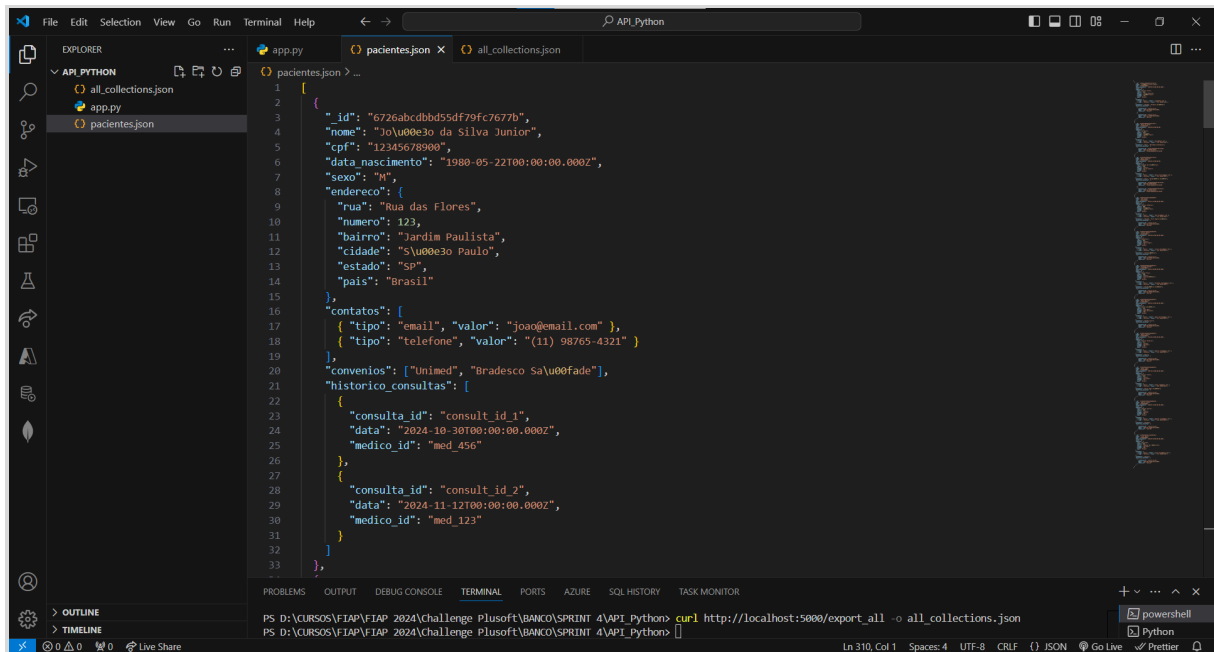


The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project named 'API\_PYTHON' with files 'app.py' and 'pacientes.json'. The main editor displays the code for 'app.py', which includes a function 'def get\_document(collection\_name, doc\_id):'. The code handles document retrieval and error cases. The Terminal pane at the bottom shows a command being executed: 'curl http://localhost:5000/export/pacientes -o pacientes.json'. The status bar at the bottom indicates the current file is 'app.py' at line 94, column 63, using UTF-8 encoding and CRLF line endings.

```
def get_document(collection_name, doc_id):
    70     if document:
    71         return jsonify(serialize_doc(document)), 200
    72     else:
    73         return jsonify({"error": "Document not found"}), 404
    74     except (TypeError):
    75         return jsonify({"error": "Invalid ID format"}), 400
    76     else:
    77         return jsonify({"error": "Collection not found"}), 404
    78
```

```
PS D:\CURSOS\FIAP\FIAP 2024\Challenge Plusoft\BANCO\SPRINT 4\API_Python> curl http://localhost:5000/export/pacientes -o pacientes.json
PS D:\CURSOS\FIAP\FIAP 2024\Challenge Plusoft\BANCO\SPRINT 4\API_Python>
```

## Arquivo JSON:



The screenshot shows the Visual Studio Code interface with the 'pacientes.json' file open in the main editor. The JSON file contains a list of patient records. The Explorer pane on the left shows the project structure with 'all\_collections.json', 'app.py', and 'pacientes.json'. The Terminal pane at the bottom shows a command being executed: 'curl http://localhost:5000/export\_all -o all\_collections.json'. The status bar at the bottom indicates the current file is 'pacientes.json' at line 310, column 1, using UTF-8 encoding and CRLF line endings.

```
{
  1  "id": "6726abcbd5d5df79fc7677b",
  2  "nome": "Jo\u00e3o da Silva Junior",
  3  "cpf": "12345678900",
  4  "data_nascimento": "1980-05-22T00:00:00.000Z",
  5  "sexo": "M",
  6  "endereco": {
  7    "rua": "Rua das Flores",
  8    "numero": 123,
  9    "bairro": "Jardim Paulista",
 10    "cidade": "S\u00e3o Paulo",
 11    "estado": "SP",
 12    "pais": "Brasil"
 13  },
 14  "contatos": [
 15    { "tipo": "email", "valor": "joao@email.com" },
 16    { "tipo": "telefone", "valor": "(11) 98765-4321" }
 17  ],
 18  "convenios": ["Unimed", "Bradesco Sa\u00fade"],
 19  "historico_consultas": [
 20    {
 21      "consulta_id": "consult id 1",
 22      "data": "2024-10-30T00:00:00.000Z",
 23      "medico_id": "med_456"
 24    },
 25    {
 26      "consulta_id": "consult id 2",
 27      "data": "2024-11-12T00:00:00.000Z",
 28      "medico_id": "med_123"
 29    }
 30  ]
 31 },
 32
 33 }
```

```
PS D:\CURSOS\FIAP\FIAP 2024\Challenge Plusoft\BANCO\SPRINT 4\API_Python> curl http://localhost:5000/export_all -o all_collections.json
PS D:\CURSOS\FIAP\FIAP 2024\Challenge Plusoft\BANCO\SPRINT 4\API_Python>
```

**curl** http://localhost:5000/export\_all -o all\_collections.json

The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project named 'API PYTHON' with files 'all\_collections.json', 'app.py', and 'pacientes.json'. The main editor shows 'app.py' with a function `def get_document(collection_name, doc_id):` that checks for document existence and returns JSON responses. The Terminal pane at the bottom shows the command `curl http://localhost:5000/export_all -o all_collections.json` being executed in a PowerShell session.

```
app.py
63 def get_document(collection_name, doc_id):
64     if document:
65         return jsonify(serialize_doc(document)), 200
66     else:
67         return jsonify({"error": "Document not found"}), 404
68     except (TypeError):
69         return jsonify({"error": "Invalid ID format"}), 400
70     else:
71         return jsonify({"error": "Collection not found"}), 404
72
73 # Update Document by ID
74
```

```
PS D:\CURSOS\FIAP\FIAP 2024\Challenge Plusoft\BANCO\SPRINT 4\API_Python> curl http://localhost:5000/export_all -o all_collections.json
PS D:\CURSOS\FIAP\FIAP 2024\Challenge Plusoft\BANCO\SPRINT 4\API_Python>
```

## Arquivo JSON:

The screenshot shows the Visual Studio Code interface with the file 'all\_collections.json' open in the editor. The file contains a JSON array of three consultation records. The Explorer pane shows the file structure. The Terminal pane at the bottom shows the same curl command as in the previous screenshot.

```
all_collections.json
1 {
2   "consultas": [
3     {
4       "id": "6726ac65bdd55df79fc76782",
5       "paciente_id": "6726abcbdd55df79fc7677b",
6       "medico_id": "6726ac30bdd55df79fc7677f",
7       "data": "2024-10-30",
8       "horario": "14:00",
9       "procedimentos": ["6726ad63bdd55df79fc7679b", "6726ad70bdd55df79fc7679c"],
10      "exames": ["6726ac90bdd55df79fc76785"],
11      "motivo": "Consulta de rotina",
12      "observacoes": "Paciente relatou dor de cabe\u00e7a constante.",
13      "status": "Conclu\u00eddo"
14    },
15    {
16      "id": "6726ac76bdd55df79fc76783",
17      "paciente_id": "6726abf9bdd55df79fc7677c",
18      "medico_id": "6726ac4dbdd55df79fc76780",
19      "data": "2024-11-12",
20      "horario": "12:00",
21      "procedimentos": ["6726ad7ebdd55df79fc7679d"],
22      "exames": ["6726aca5bdd55df79fc76786"],
23      "motivo": "Consulta de retorno",
24      "observacoes": "Paciente relatou dor na regi\u00e3o tor\u00e1xica.",
25      "status": "Conclu\u00eddo"
26    },
27    {
28      "id": "6726b201bdd55df79fc767e5",
29      "paciente_id": "6726ac0cbdd55df79fc7677d",
30      "medico_id": "6726ac4dbdd55df79fc76780",
31      "data": "2024-11-12",
32      "horario": "12:00",
33      "procedimentos": [],
34    }
35  ]
36 }
```

```
PS D:\CURSOS\FIAP\FIAP 2024\Challenge Plusoft\BANCO\SPRINT 4\API_Python> curl http://localhost:5000/export_all -o all_collections.json
PS D:\CURSOS\FIAP\FIAP 2024\Challenge Plusoft\BANCO\SPRINT 4\API_Python>
```