

FIAP

LARISSA ARAÚJO GAMA ALVARENGA – 96496 - 2TDSPS

LARISSA LOPES OLIVEIRA – 552628 - 2TDSPB

LUNA FAUSTINO LIMA – 552473 - 2TDSPB

SPRINT 1 – JAVA ADVANCED

São Paulo

2024

EXPLICAÇÃO DA SOLUÇÃO

Criamos um programa que realizará o cadastro de Pacientes, Dentistas e Clínicas no banco de dados Oracle. Para Pacientes e Clínicas ainda é possível adicionar Endereços.

LINK VÍDEO YOUTUBE

https://youtu.be/QeDd_QnXaxM?si=Sq9gw9hwt4qJPtOO

LINK REPOSITÓRIO

<https://github.com/LunaFaustino/Sprint1-JavaAdvanced.git>

CRONOGRAMA DE DESENVOLVIMENTO DA SPRINT 1 (ATÉ 07/10)

Responsável pela Sprint 1 de **COMPLIANCE & QUALITY ASSURANCE**: Larissa Lopes;

Responsável pela Sprint 1 de **MASTERING RELATIONAL AND NON RELATIONAL DATABASE**: Larissa Araújo;

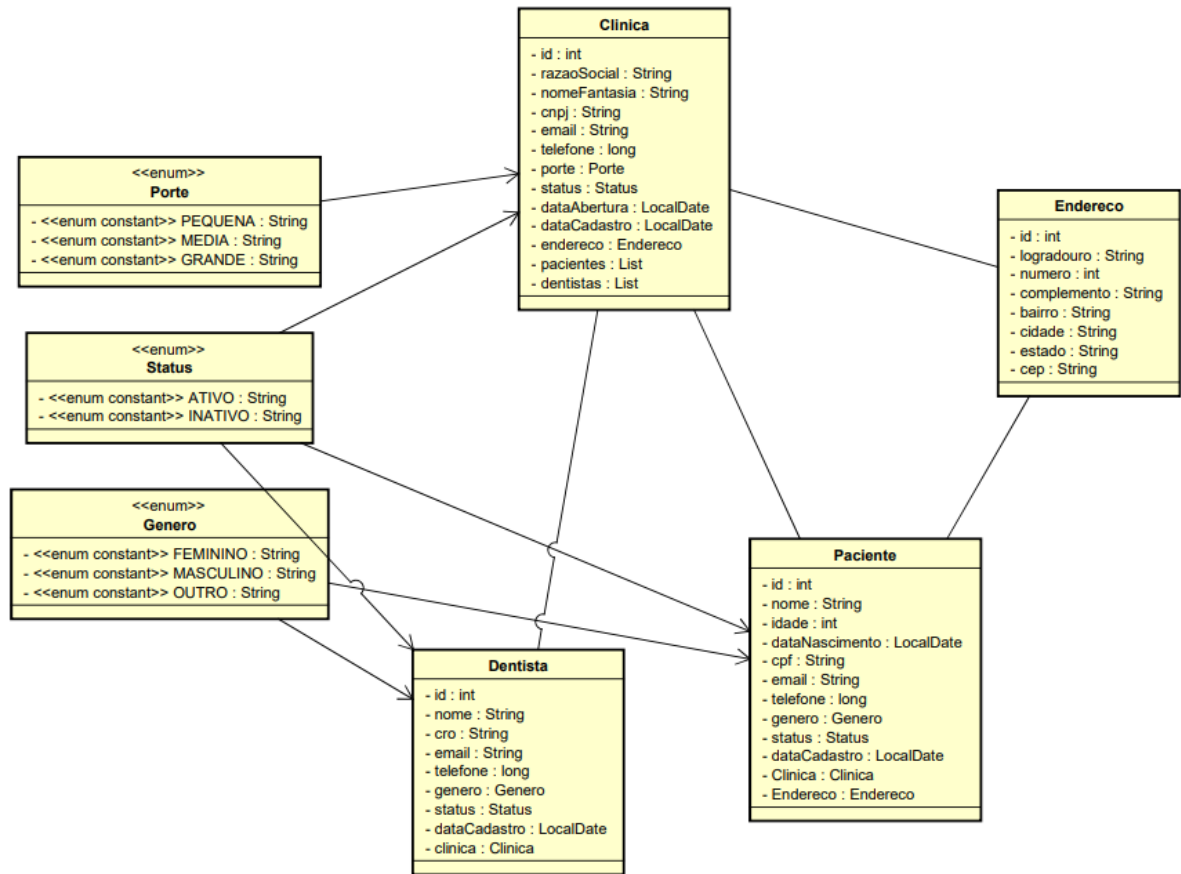
Responsável pela Sprint 1 de **DEVOPS TOOLS E CLOUD COMPUTING**: Larissa Araújo;

Responsável pela Sprint 1 de **ADVANCED BUSINESS DEVELOPMENT WITH .NET**: Larissa Araújo;

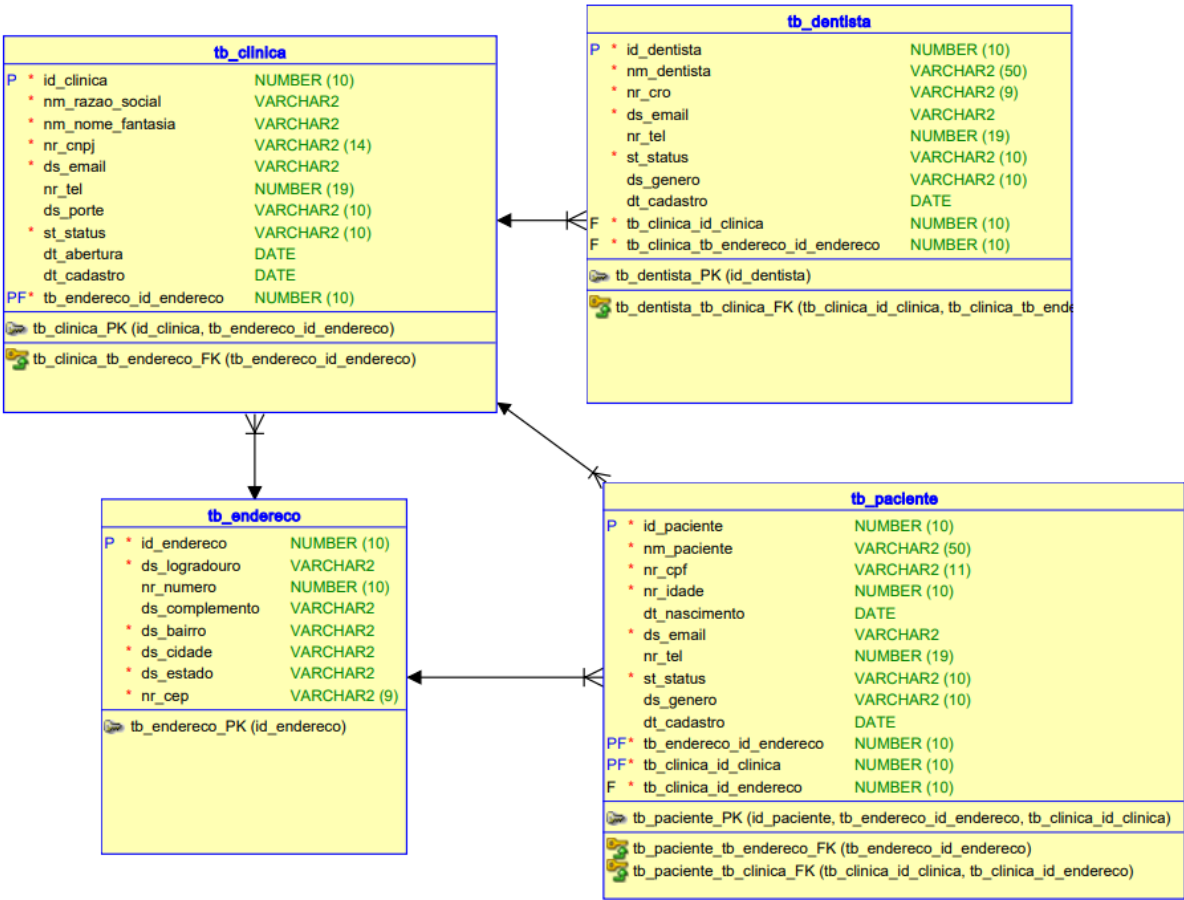
Responsável pela Sprint 1 de **DISRUPTIVE ARCHITECTURES: IOT, IOB & GENERATIVE IA**: Luna;

Responsável pela Sprint 1 de **JAVA ADVANCED**: Luna;

DIAGRAMA DE CLASSES



DER



ENDPOINTS

- POSTs:

/clinicas – Para adicionar uma clínica a tabela tb_clinica; Exemplo de JSON:

```
{
  "razaoSocial": "Clínica Bem Estar Ltda",
  "nomeFantasia": "Clínica Bem Estar",
  "cnpj": "12345678000190",
  "email": "contato@clinicabemestar.com",
  "telefone": 1122334455,
  "porte": "MEDIA",
  "status": "ATIVO",
  "dataAbertura": "2020-01-15",
  "endereco": {
    "logradouro": "Avenida Central",
    "numero": 1000,
    "complemento": "Sala 200",
    "bairro": "Centro",
    "cidade": "São Paulo",
    "estado": "SP",
    "cep": "01000-000"
  }
}
```

/dentistas – Para adicionar um dentista a tabela tb_dentista; Exemplo de JSON:

```
{
  "nome": "Dra. Ana Pereira",
  "cro": "SP123456",
  "email": "ana.pereira@exemplo.com",
  "telefone": 11987654321,
  "genero": "FEMININO",
  "status": "ATIVO",
  "clinicald": 1
}
```

/pacientes – Para adicionar um paciente a tabela tb_paciente; Exemplo de JSON:

```
{
  "nome": "Carlos Silva",
  "idade": 35,
  "dataNascimento": "1986-08-12",
  "cpf": "12345678901",
  "email": "carlos.silva@exemplo.com",
  "telefone": 11999998888,
  "genero": "MASCULINO",
  "status": "ATIVO",
  "clinicald": 1,
  "endereco": {
    "logradouro": "Rua das Palmeiras",
    "numero": 500,
    "complemento": "Apto 45",
    "bairro": "Jardim Paulista",
    "cidade": "São Paulo",
    "estado": "SP",
    "cep": "01311-000"
  }
}
```

- PUTs

/clinicas/{id} – Atualiza os dados da clínica desse Id; Exemplo de JSON:

```
{  
  "razaoSocial": "Clínica Saúde Total Ltda",  
  "nomeFantasia": "Clínica Saúde Total",  
  "cnpj": "12345678000190",  
  "email": "contato@saudetotal.com",  
  "telefone": "11987654321",  
  "porte": "GRANDE",  
  "status": "ATIVO",  
  "dataAbertura": "2020-01-15"  
}
```

/dentistas/{id} - Atualiza os dados do dentista desse Id; Exemplo de JSON:

```
{  
  "nome": "Dra. Ana Luisa Pereira",  
  "cro": "SP654321",  
  "email": "ana.luisa@exemplo.com",  
  "telefone": "11987654444",  
  "genero": "FEMININO",  
  "status": "ATIVO",  
  "clinicald": 1  
}
```


/pacientes/{id} - Atualiza os dados do paciente desse Id; Exemplo de JSON:

```
{  
  "nome": "Carlos Alberto Silva",  
  "idade": 36,  
  "dataNascimento": "1986-08-12",  
  "cpf": "12345678901",  
  "email": "carlos.alberto@exemplo.com",  
  "telefone": "11999997777",  
  "genero": "MASCULINO",  
  "status": "ATIVO",  
  "clinicald": 1  
}
```

/enderecos/{id} - Atualiza os dados do endereco desse Id. Exemplo de JSON:

```
{  
  "logradouro": "Nova Rua",  
  "numero": 456,  
  "complemento": "Casa 2",  
  "bairro": "Novo Bairro",  
  "cidade": "Nova Cidade",  
  "estado": "SP",  
  "cep": "12345-678"  
}
```

- GETs:

/clinicas – Retorna todas as clínicas cadastradas no banco de dados;

/clinicas/{id} – Retorna a clínica referente ao Id específico;

/dentistas - Retorna todos os dentistas cadastrados no banco de dados;

/dentistas/{id} – Retorna o dentista referente ao Id específico;

/pacientes - Retorna todos os pacientes cadastrados no banco de dados;

/pacientes/{id} – Retorna o paciente referente ao Id específico;

/enderecos – Retorna todos os endereços cadastrados no banco de dados;

/endereços/{id} – Retorna o endereço referente ao Id específico.

- DELETES:

/clinicas/{id} – Deleta a clínica relacionada ao Id específico (os dentistas, pacientes e endereço também serão apagados);

/dentistas/{id} – Deleta o dentista relacionado ao Id específico;

/pacientes/{id} – Deleta o paciente relacionado ao Id específico (o seu endereço também é deletado);

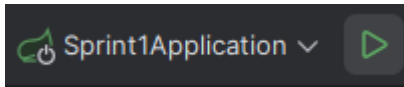
PS¹: O endereço não tem método POST próprio, então é preciso criar uma clínica ou paciente para então vincular um endereço. Também não tem método DELETE próprio, então ele é excluído quando o seu “dono” é excluído.

PS²: Pacientes e dentistas são vinculados a uma clínica, então caso a clínica seja excluída, todos os pacientes e dentistas vinculados a ela também serão excluídos.

ORIENTAÇÃO DE COMO RODAR A APLICAÇÃO

Para rodar a aplicação é preciso de um programa que consiga fazer os métodos do RESTFUL, aqui usaremos como exemplo o POSTMAN.

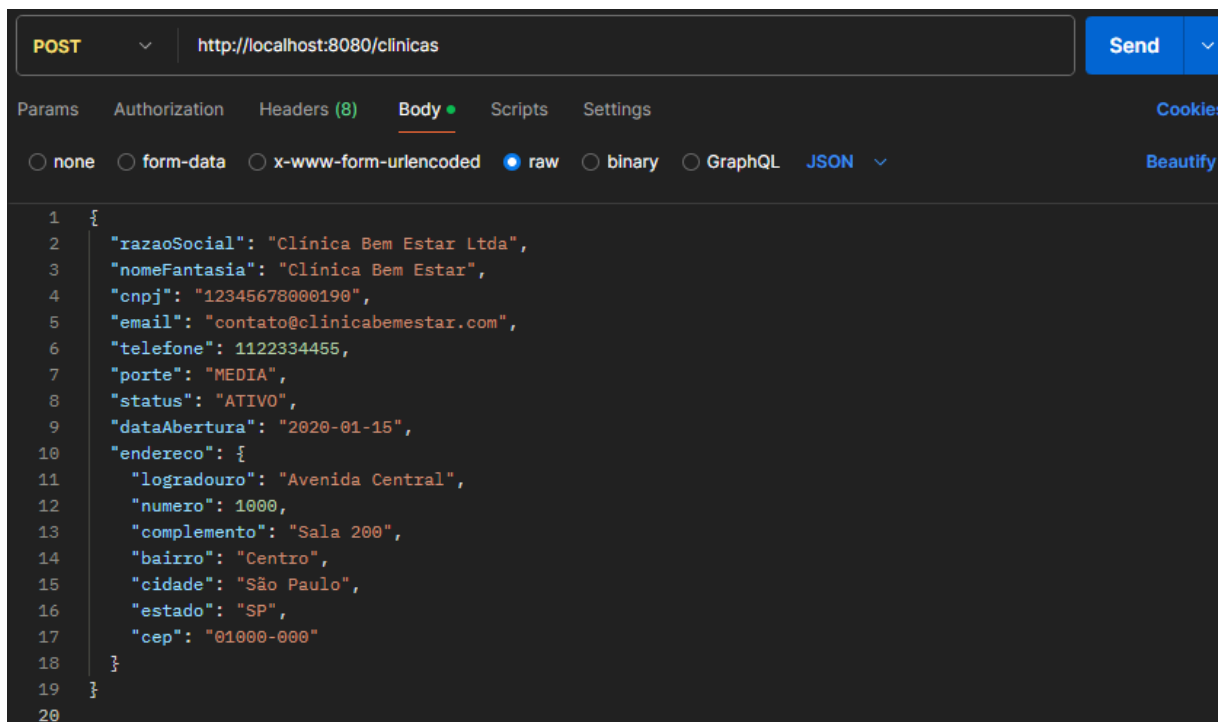
- 1- Abra o projeto na IDE desejada e clique no botão de “RUN” no topo da IDE ou dentro da classe Sprint1Application;



- 2- Se atente ao console da IDE, assim que tiver rodado com sucesso, aparecerá um aviso informando que a aplicação já está rodando em uma porta (por padrão é a 8080);

```
: LiveReload server is running on port 35729
: Tomcat started on port 8080 (http) with context path '/'
: Started Sprint1Application in 9.686 seconds (process running)
```

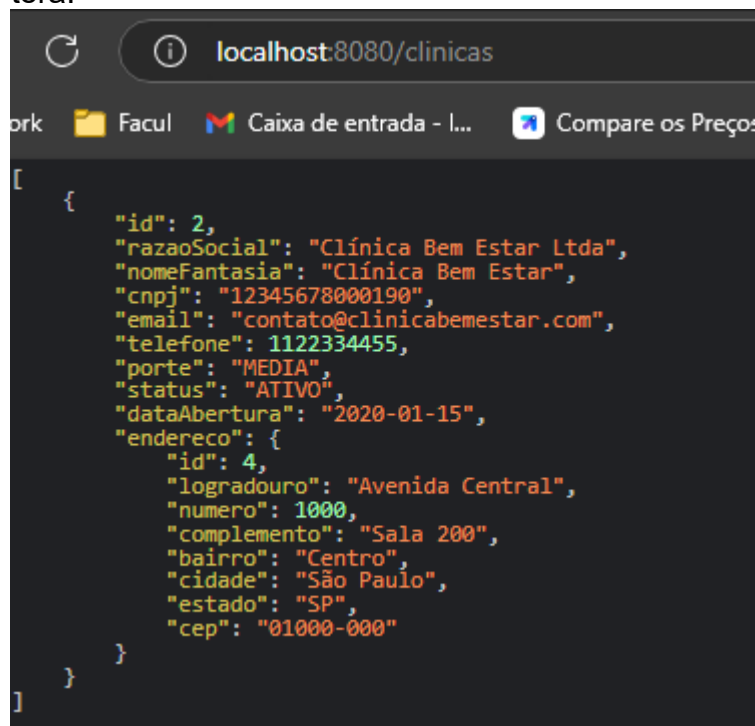
- 3- Abra o programa para fazer a requisição (no nosso exemplo o POSTMAN), selecione o método desejado, no caso o POST e então a URL <http://localhost:8080> + o endpoint, sendo /clinicas a do exemplo. Além disso vai até a parte de “Body” e então a opção de “raw” do programa e cole o JSON desejado e então clique em “SEND”:



4- Caso dê tudo certo, o programa irá retornar o código 201 e o JSON cadastrado.

```
1  {
2    "id": 2,
3    "razaoSocial": "Clínica Bem Estar Ltda",
4    "nomeFantasia": "Clínica Bem Estar",
5    "cnpj": "12345678000190",
6    "email": "contato@clinicabemestar.com",
7    "telefone": 1122334455,
8    "porte": "MEDIA",
9    "status": "ATIVO",
10   "dataAbertura": "2020-01-15",
11   "endereco": {
12     "id": 4,
13     "logradouro": "Avenida Central",
14     "numero": 1000,
15     "complemento": "Sala 200",
16     "bairro": "Centro",
17     "cidade": "São Paulo",
18     "estado": "SP",
19     "cep": "01000-000"
20   }
21 }
```

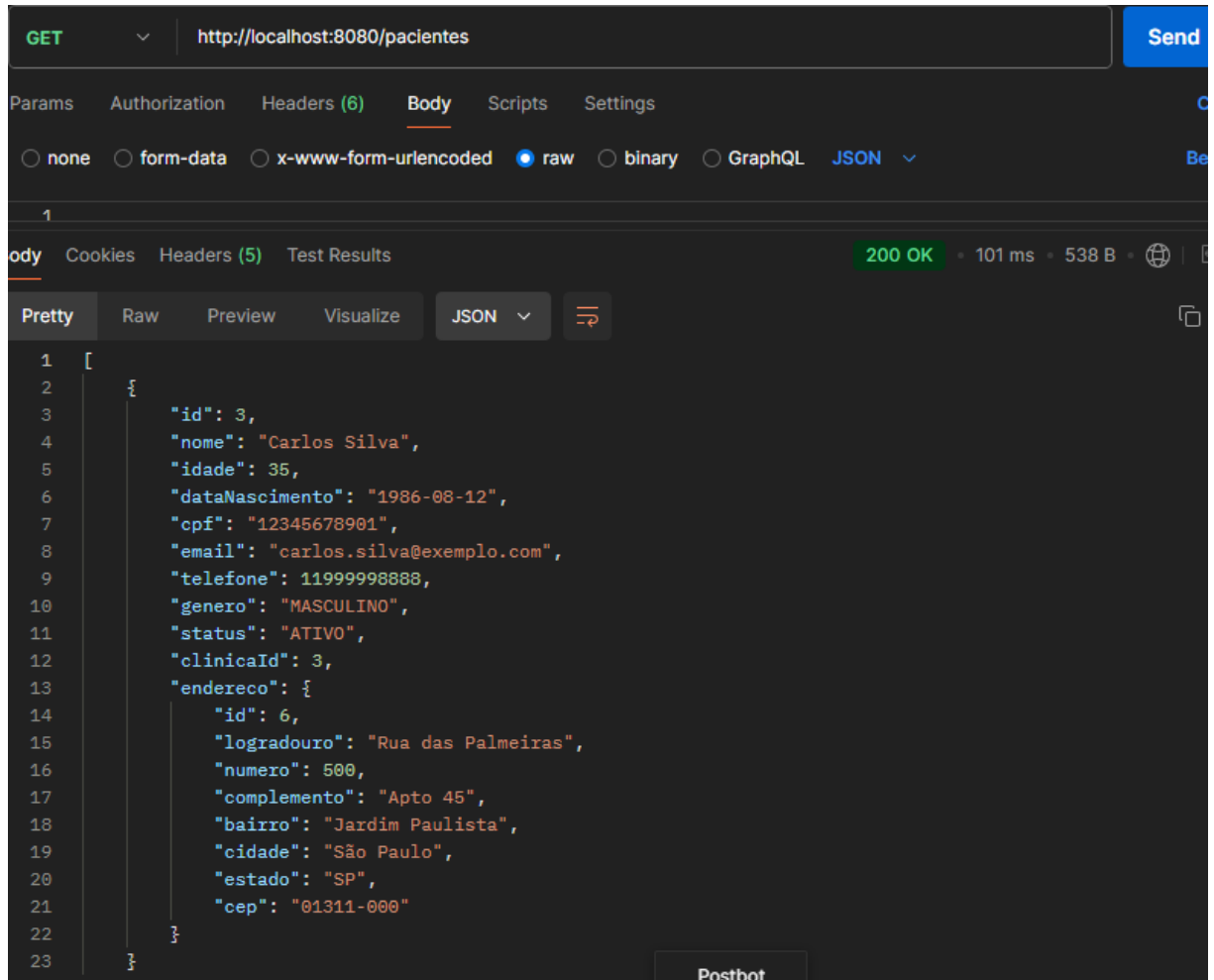
5- Pelo navegador é possível fazer o método GET e verificar se foi feita a inserção com sucesso, só colocar na barra de endereço o localhost:8080/clinicas que terá:



The screenshot shows a web browser window with the address bar displaying `localhost:8080/clinicas`. The browser's address bar includes a refresh button, an information icon, and the address text. Below the address bar, there are several tabs: "ork", "Facul", "Caixa de entrada - l...", and "Compare os Preços". The main content area of the browser displays a JSON array containing one object, which matches the JSON shown in the previous code block. The JSON is formatted with syntax highlighting, showing the nested structure of the clinic and its address.

```
[
  {
    "id": 2,
    "razaoSocial": "Clínica Bem Estar Ltda",
    "nomeFantasia": "Clínica Bem Estar",
    "cnpj": "12345678000190",
    "email": "contato@clinicabemestar.com",
    "telefone": 1122334455,
    "porte": "MEDIA",
    "status": "ATIVO",
    "dataAbertura": "2020-01-15",
    "endereco": {
      "id": 4,
      "logradouro": "Avenida Central",
      "numero": 1000,
      "complemento": "Sala 200",
      "bairro": "Centro",
      "cidade": "São Paulo",
      "estado": "SP",
      "cep": "01000-000"
    }
  }
]
```

- 6- Agora vamos usar o método PUT para atualizar o endereço de um paciente, primeiro usamos o GET com o endpoint /pacientes para verificar os cadastros e pegar o Id do paciente que queremos atualizar.



```
GET http://localhost:8080/pacientes Send

Params Authorization Headers (6) Body Scripts Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
200 OK 101 ms 538 B

Pretty Raw Preview Visualize JSON
1 [
2   {
3     "id": 3,
4     "nome": "Carlos Silva",
5     "idade": 35,
6     "dataNascimento": "1986-08-12",
7     "cpf": "12345678901",
8     "email": "carlos.silva@exemplo.com",
9     "telefone": 11999998888,
10    "genero": "MASCULINO",
11    "status": "ATIVO",
12    "clinicaId": 3,
13    "endereco": {
14      "id": 6,
15      "logradouro": "Rua das Palmeiras",
16      "numero": 500,
17      "complemento": "Apto 45",
18      "bairro": "Jardim Paulista",
19      "cidade": "São Paulo",
20      "estado": "SP",
21      "cep": "01311-000"
22    }
23  }
24 ]
```

7- Agora vamos usar o método PUT com o Id 6 e passar um novo endereço com a URL endereços/6, então irá retornar o código 200

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/enderecos/6`. The request body is a JSON object with address details. The response is a 200 OK status with a JSON object containing the updated address data, including an 'id' of 6.

Request:

```
PUT http://localhost:8080/enderecos/6
```

Request Body (JSON):

```
{
  "logradouro": "Nova Rua",
  "numero": 456,
  "complemento": "Casa 2",
  "bairro": "Novo Bairro",
  "cidade": "Nova Cidade",
  "estado": "SP",
  "cep": "12345-678"
}
```

Response: 200 OK (108 ms, 310 B)

Response Body (JSON):

```
{
  "id": 6,
  "logradouro": "Nova Rua",
  "numero": 456,
  "complemento": "Casa 2",
  "bairro": "Novo Bairro",
  "cidade": "Nova Cidade",
  "estado": "SP",
  "cep": "12345-678"
}
```

Postbot

- 8- Por último veremos um exemplo do método DELETE, onde iremos excluir a clínica de Id 3, então excluindo o restante do banco de dados, só usar o endpoint /clinicas/3 e o programa vai retornar o código 200

