

FIAP

LARISSA ARAÚJO GAMA ALVARENGA – 96496 - 2TDSPS

LARISSA LOPES OLIVEIRA – 552628 - 2TDSPB

LUNA FAUSTINO LIMA – 552473 - 2TDSPB

**SPRINT 2 – 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.

## **EVOLUÇÕES DESTA VERSÃO**

Incluimos HATEOAS, então agora os endpoints retornam outros endpoints relacionados que podem ser úteis para quem estiver utilizando a nossa API. Além disso, fizemos uso do Lombok para simplificar os códigos.

## **LINK VÍDEO YOUTUBE**

[https://youtu.be/uZyYlbf\\_XSk?si=bdhZGs2LpxR0WNrp](https://youtu.be/uZyYlbf_XSk?si=bdhZGs2LpxR0WNrp)

## **LINK REPOSITÓRIO**

<https://github.com/LunaFaustino/Sprints-JavaAdvanced>

## **CRONOGRAMA DE DESENVOLVIMENTO DA SPRINT 2 (ATÉ 08/11)**

Responsável pela Sprint 2 de **COMPLIANCE & QUALITY ASSURANCE**: Larissa Lopes;

Responsável pela Sprint 2 de **MOBILE APP DEVELOPMENT**: Larissa Lopes;

Responsável pela Sprint 2 de **MASTERING RELATIONAL AND NON RELATIONAL DATABASE**: Larissa Araújo;

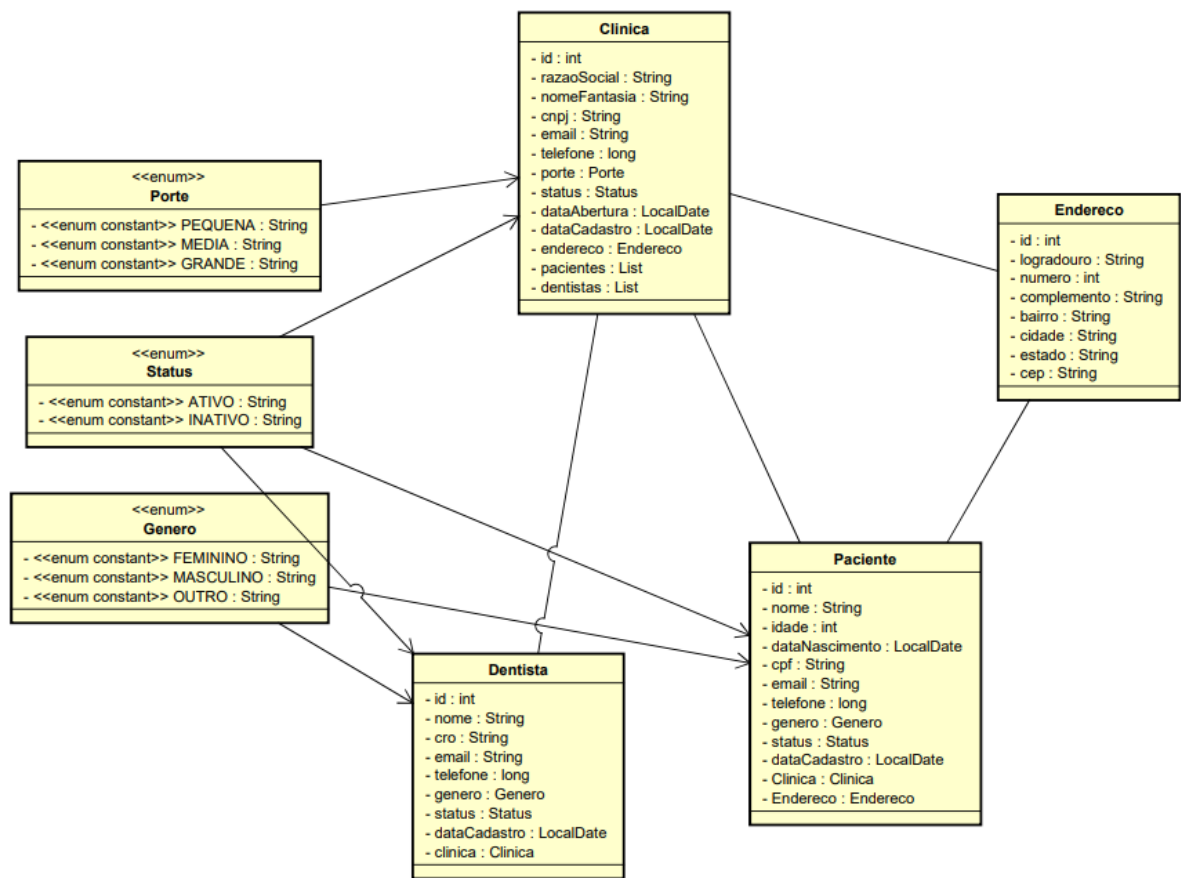
Responsável pela Sprint 2 de **ADVANCED BUSINESS DEVELOPMENT WITH .NET**: Larissa Araújo;

Responsável pela Sprint 2 de **DEVOPS TOOLS E CLOUD COMPUTING**: Larissa Araújo e Luna;

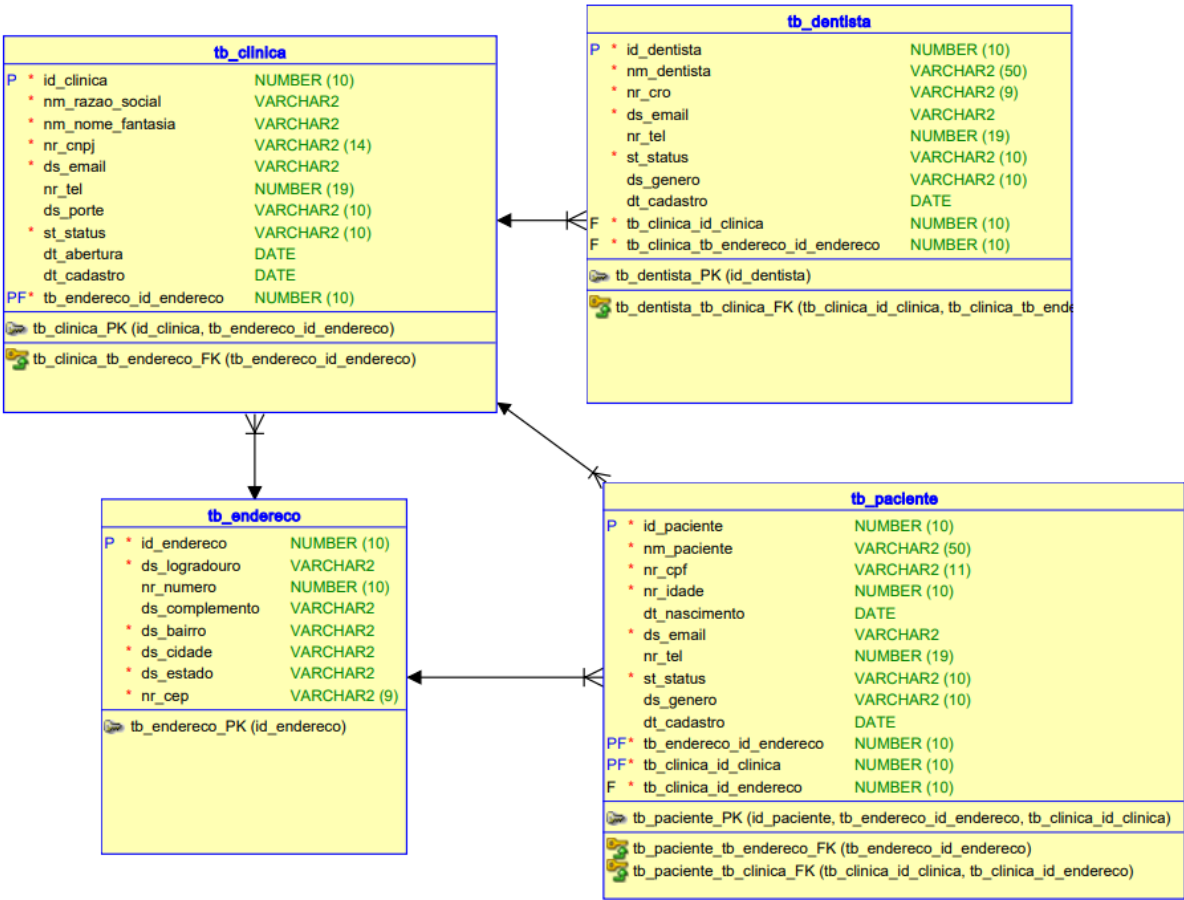
Responsável pela Sprint 2 de **DISRUPTIVE ARCHITECTURES: IOT, IOB & GENERATIVE IA**: Luna;

Responsável pela Sprint 2 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);

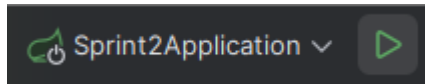
**PS<sup>1</sup>:** O endereço não tem método POST próprio, então é preciso criar uma clínica ou paciente para então vincular um endereço. Também não tem método DELETE próprio, então ele é excluído quando o seu “dono” é excluído.

**PS<sup>2</sup>:** 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.

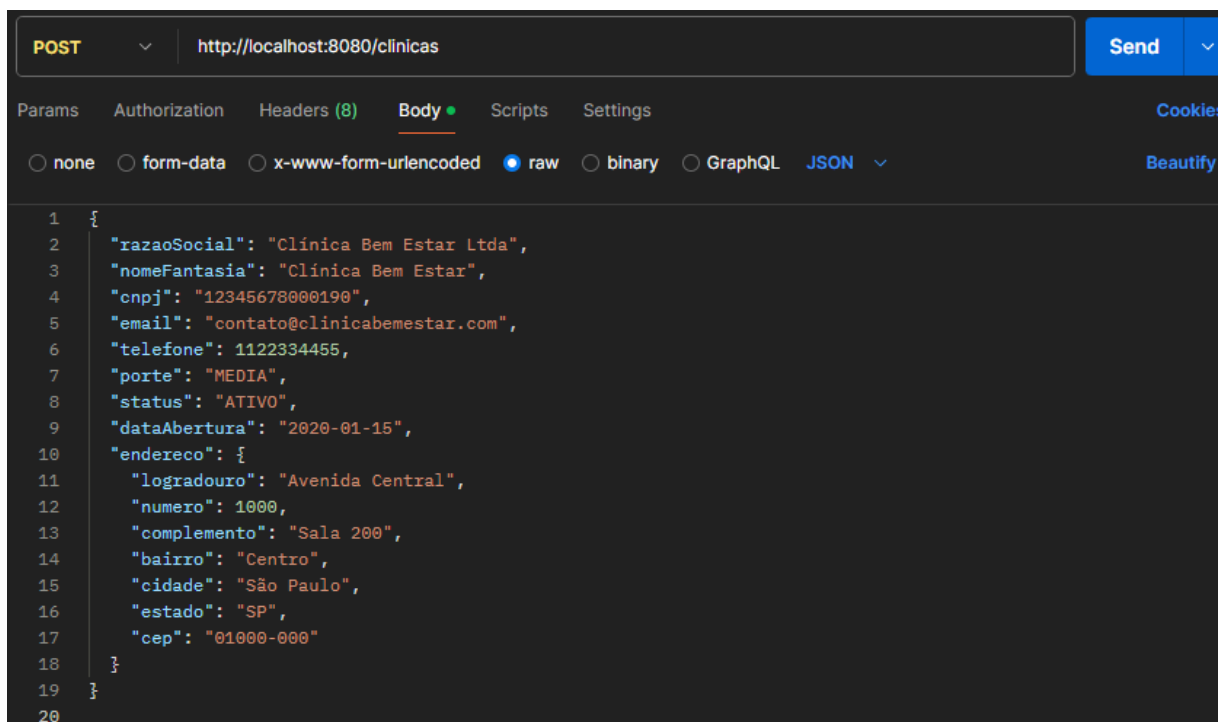
- 1- Abra o projeto na IDE desejada e clique no botão de “RUN” no topo da IDE ou dentro da classe Sprint1Application;



- 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.

```
Body ▾ 201 Created • 349 ms • 684 B • 🌐 | 📄 ⋮
```

```
Pretty Raw Preview Visualize JSON ▾ 🔁 📄 🔍
```

```
1  {
2      "id": 1,
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": 1,
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     "_links": {
22         "self": {
23             "href": "http://localhost:8080/clinicas/1"
24         },
25         "todasClinicas": {
26             "href": "http://localhost:8080/clinicas"
27         }
28     }
29 }
```

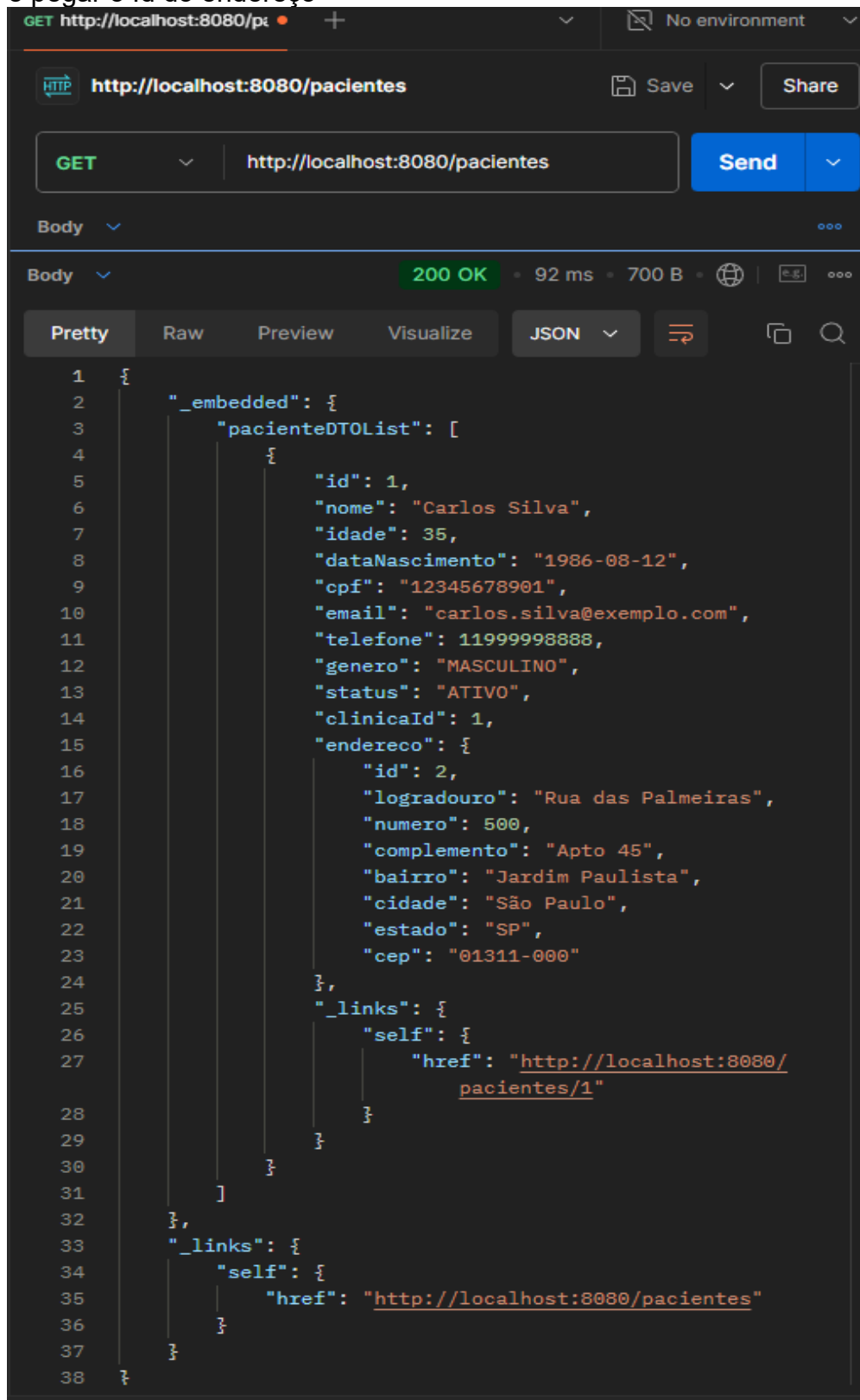
- 5- Pelo navegador é possível fazer o método GET e verificar se foi feito a inserção com sucesso, só colocar na barra de endereço o [localhost:8080/clinicas](http://localhost:8080/clinicas) que terá:



The image shows a web browser window with the address bar displaying `localhost:8080/clinicas`. The main content area shows a JSON response, which is a list of clinic objects. The first object in the list is for a clinic with ID 1, named 'Clínica Bem Estar Ltda', located at 'Avenida Central, 1000, Sala 200' in 'Centro', 'São Paulo', 'SP', with CEP '01000-000'. The JSON is formatted with syntax highlighting.

```
{
  "_embedded": {
    "clinicaDTOList": [
      {
        "id": 1,
        "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": 1,
          "logradouro": "Avenida Central",
          "numero": 1000,
          "complemento": "Sala 200",
          "bairro": "Centro",
          "cidade": "São Paulo",
          "estado": "SP",
          "cep": "01000-000"
        },
        "_links": {
          "self": {
            "href": "http://localhost:8080/clinicas/1"
          }
        }
      }
    ]
  },
  "_links": {
    "self": {
      "href": "http://localhost:8080/clinicas"
    }
  }
}
```

- 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 endereço



The screenshot shows a REST client interface with a GET request to `http://localhost:8080/pacientes` and its corresponding JSON response. The response status is `200 OK` with a response time of `92 ms` and a body size of `700 B`. The JSON body is formatted in the 'Pretty' view and contains a list of patients with their details and a self-link.

```
1  {
2    "_embedded": {
3      "pacienteDTOList": [
4        {
5          "id": 1,
6          "nome": "Carlos Silva",
7          "idade": 35,
8          "dataNascimento": "1986-08-12",
9          "cpf": "12345678901",
10         "email": "carlos.silva@exemplo.com",
11         "telefone": 11999998888,
12         "genero": "MASCULINO",
13         "status": "ATIVO",
14         "clinicaId": 1,
15         "endereco": {
16           "id": 2,
17           "logradouro": "Rua das Palmeiras",
18           "numero": 500,
19           "complemento": "Apto 45",
20           "bairro": "Jardim Paulista",
21           "cidade": "São Paulo",
22           "estado": "SP",
23           "cep": "01311-000"
24         },
25         "_links": {
26           "self": {
27             "href": "http://localhost:8080/pacientes/1"
28           }
29         }
30       }
31     ],
32     "_links": {
33       "self": {
34         "href": "http://localhost:8080/pacientes"
35       }
36     }
37   }
38 }
```

- 7- Agora vamos usar o método PUT com o Id 2 e passar um novo endereço com a URL `enderecos/6`, então irá retornar o código 200 indicando que deu certo

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/enderecos/2`. The request body is a JSON object with address details. The response is a 200 OK status with a JSON body containing the updated address and links.

**Request:**

```
PUT http://localhost:8080/enderecos/2
```

Params Authorization Headers (8) **Body** Scripts Settings

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

```
1 {
2   "logradouro": "Nova Rua",
3   "numero": 456,
4   "complemento": "Casa 2",
5   "bairro": "Novo Bairro",
6   "cidade": "Nova Cidade",
7   "estado": "SP",
8   "cep": "12345-678"
9 }
10
```

**Response:**

**Body** Cookies Headers (5) Test Results **200 OK**

Pretty Raw Preview Visualize **JSON** ▾

```
1 {
2   "id": 2,
3   "logradouro": "Nova Rua",
4   "numero": 456,
5   "complemento": "Casa 2",
6   "bairro": "Novo Bairro",
7   "cidade": "Nova Cidade",
8   "estado": "SP",
9   "cep": "12345-678",
10  "_links": {
11    "self": {
12      "href": "http://localhost:8080/enderecos/2"
13    },
14    "todosEnderecos": {
15      "href": "http://localhost:8080/enderecos"
16    }
17  }
18 }
```

- 8- Por último veremos um exemplo do método DELETE, onde iremos excluir a clínica de Id 1, então excluindo o restante do banco de dados, só usar o endpoint /clinicas/1 e o programa vai retornar o código 204, indicando que deu certo.

