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**

**Diagrama

Descrição gerada automaticamente**

**DER**

**Diagrama

Descrição gerada automaticamente**

**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",

  "clinicaId": 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",

  "clinicaId": 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",

"clinicaId": 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",

"clinicaId": 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;



1. 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);

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

1. 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”:

Texto

Descrição gerada automaticamente

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

Texto

Descrição gerada automaticamente

1. 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:  
   Texto

   Descrição gerada automaticamente
2. 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.

Texto

Descrição gerada automaticamente

1. 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

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

1. 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

Tela de celular com aplicativo aberto

Descrição gerada automaticamente