

Documentação Técnica

Sprint 4

API RESTful

|Evellyn Barbosa Ferreira |RM562744
|Henrique Sinkevicius Maran |RM562977
|Pedro Henrique Crus Lemos |RM565605

Link do repositório: <https://github.com/PedroCLH2/Sprint-java-4.git>

-----Sumário

1. Objetivo e Escopo do Projeto
2. Descrição das Funcionalidades
3. Tabela de Endpoints (API RESTful)
4. Protótipo (Prints da API)
5. Modelo de Entidade-Relacionamento (MER)
6. Diagrama de Classes (Atualizado)

-----1. Objetivo e Escopo do Projeto

O objetivo desta entrega foi evoluir o Sistema de Gestão de Clínica Médica para uma arquitetura moderna e distribuída, implementando uma **API RESTful** robusta utilizando Java com o framework **Quarkus**.

O escopo do projeto abrange o gerenciamento completo (CRUD) das entidades principais: **Pacientes**, **Médicos** e **Consultas**. A API foi projetada para ser consumida por aplicações front-end, oferecendo *endpoints* padronizados, validação de dados de entrada (via *Bean Validation*), transferência eficiente de dados (via DTOs) e tratamento global de erros para garantir respostas HTTP consistentes.

----2. Descrição das Funcionalidades

A API oferece as seguintes funcionalidades principais:

- **Gestão de Pacientes:** Cadastro, listagem, busca detalhada por ID, atualização de dados cadastrais e remoção de pacientes.
- **Gestão de Médicos:** Cadastro completo com CRM e especialidade, além das operações de busca, atualização e remoção.
- **Agendamento de Consultas:** Funcionalidade que integra médicos e pacientes, permitindo agendar novas consultas com validação de datas futuras e listar os agendamentos realizados com os dados completos das partes envolvidas.
- **Validação de Dados:** Garantia de integridade impedindo cadastro de campos obrigatórios vazios, e-mails inválidos ou datas no passado.

-----3. Tabela de Endpoints (API RESTful)

Recurso	Método	URI (Caminho)	Descrição	Status Sucesso	Status Erro
Paciente	GET	/paci...	Lista todos os pacientes	200 OK	500 Int...
	POST	/paci...	Cadastra novo paciente	201 Cr...	400 B...
	GET	/paci...	Busca paciente por ID	200 OK	404 N...
	PUT	/paci...	Atualiza dados do paciente	200 OK	404 N...
	DELETE	/paci...	Remove um paciente	204 N...	404 N...
Médico	GET	/medi...	Lista todos os médicos	200 OK	500 Int...
	POST	/medi...	Cadastra novo médico	201 Cr...	400 B...
	GET	/medi...	Busca médico por ID	200 OK	404 N...
	PUT	/medi...	Atualiza dados do médico	200 OK	404 N...
	DELETE	/medi...	Remove um médico	204 N...	404 N...
Consulta	GET	/cons...	Lista consultas (com detalhes)	200 OK	500 Int...
	POST	/cons...	Agenda nova consulta	201 Cr...	400 B...

-----4. Protótipo (Prints da API)

A seguir, evidências de funcionamento dos principais *endpoints* da API.

Figura 1 - Cadastro de Paciente (POST) com Sucesso (201 Created)

The screenshot shows a REST API testing interface with the following details:

- Method:** POST
- URL:** {{base_url}}/pacientes
- Body:** Raw JSON (selected)

```
1 {
2   "nome": "Mariana Souza",
3   "email": "mariana.souza@email.com",
4   "telefone": "11987654321"
5 }
```
- Response Headers:** 201 Created, 6.03 s, 442 B
- Response Body:** JSON

```
1 {
2   "id": 2,
3   "nome": "Mariana Souza",
4   "email": "mariana.souza@email.com",
5   "telefone": "11987654321"
6 }
```

Figura 2 - Validação de Dados Incorretos (400 Bad Request)

The screenshot shows the Postman interface with a POST request to `{{base_url}}/pacientes`. The request body is set to "raw" and contains the following JSON:

```

1 {
2   "name": "",
3   "email": "email-invalido",
4   "telefone": "123"
5 }

```

The response status is 400 Bad Request, with the following error message in the body:

```

1 {
2   "title": "Constraint Violation",
3   "status": 400,
4   "violations": [
5     {
6       "field": "cadastrar.dto.nome",
7       "message": "O nome deve ter entre 3 e 100 caracteres"
8     },
9     {
10       "field": "cadastrar.dto.telefone",
11       "message": "O telefone deve conter 10 ou 11 dígitos numéricos"
12     },
13     {
14       "field": "cadastrar.dto.email",
15       "message": "O e-mail deve ser válido"
16     },
17     {
18       "field": "cadastrar.dto.nome",
19     }
20 }

```

Figura 3 - Listagem de Consultas com Relacionamento (GET)

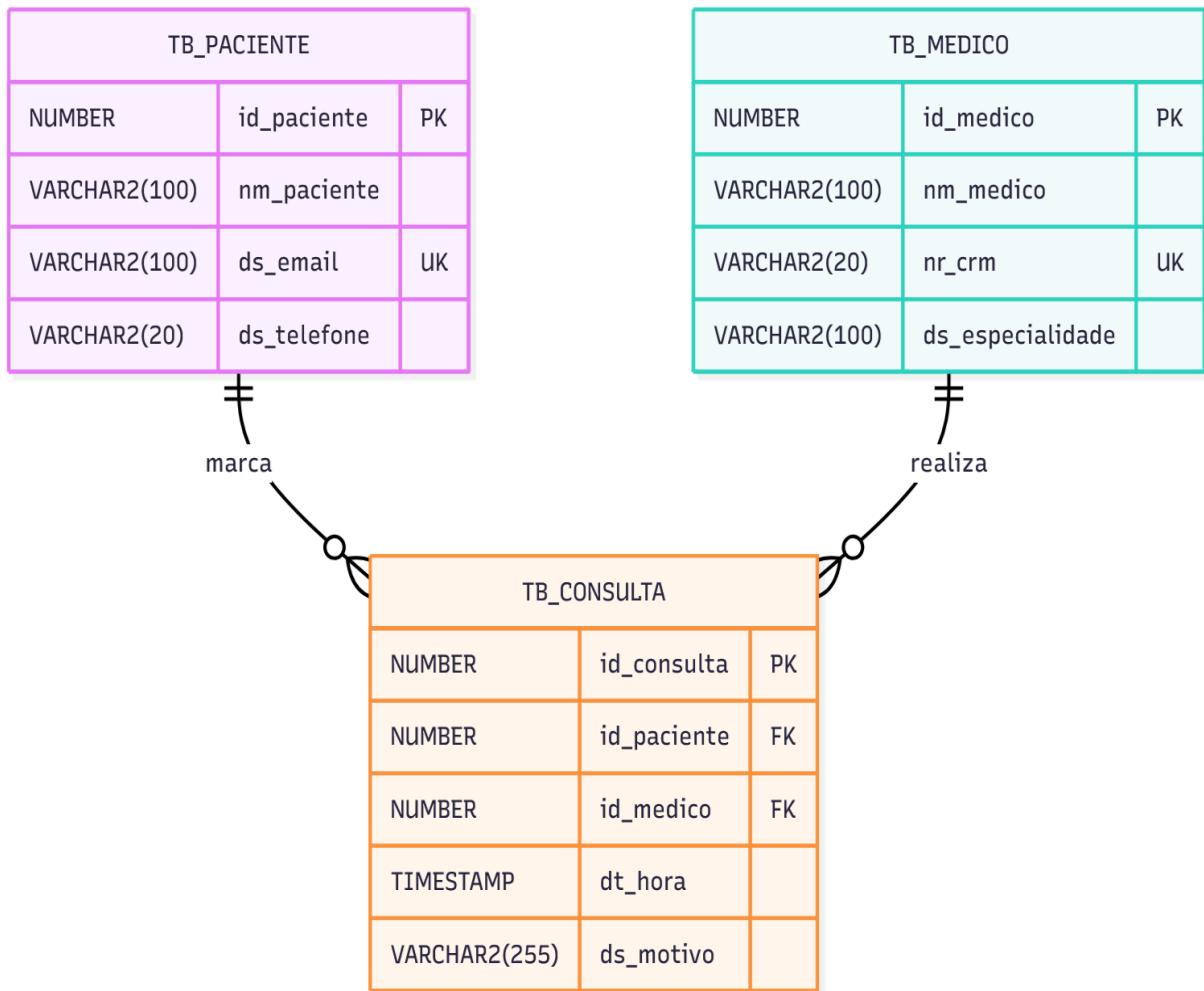
The screenshot shows the Postman interface with a GET request to `{{base_url}}/consultas`. The request body is set to "none". The response status is 200 OK, with the following JSON body:

```

1 [
2   {
3     "id": 2,
4     "paciente": [
5       {
6         "id": 1,
7         "name": "Thiago Silva",
8         "email": null,
9         "telefone": null
10      },
11      {
12        "medico": [
13          {
14            "id": 1,
15            "name": "Dr. Gregory House",
16            "crm": null,
17            "especialidade": null
18          }
19        ],
20        "dataHora": "2025-12-20T15:30:00",
21        "motivo": "Door de cabeça constante"
22      }
23    ]
24  }
25 ]

```

-----5. Modelo de Entidade-Relacionamento (MER)



-----6. Diagrama de Classes

