

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ**  
**CAMPUS MARACANAÚ**  
**CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**YAN PEDRO FAÇANHA BRASILEIRO**

**Relatório Técnico - Sistema SIGEJ**

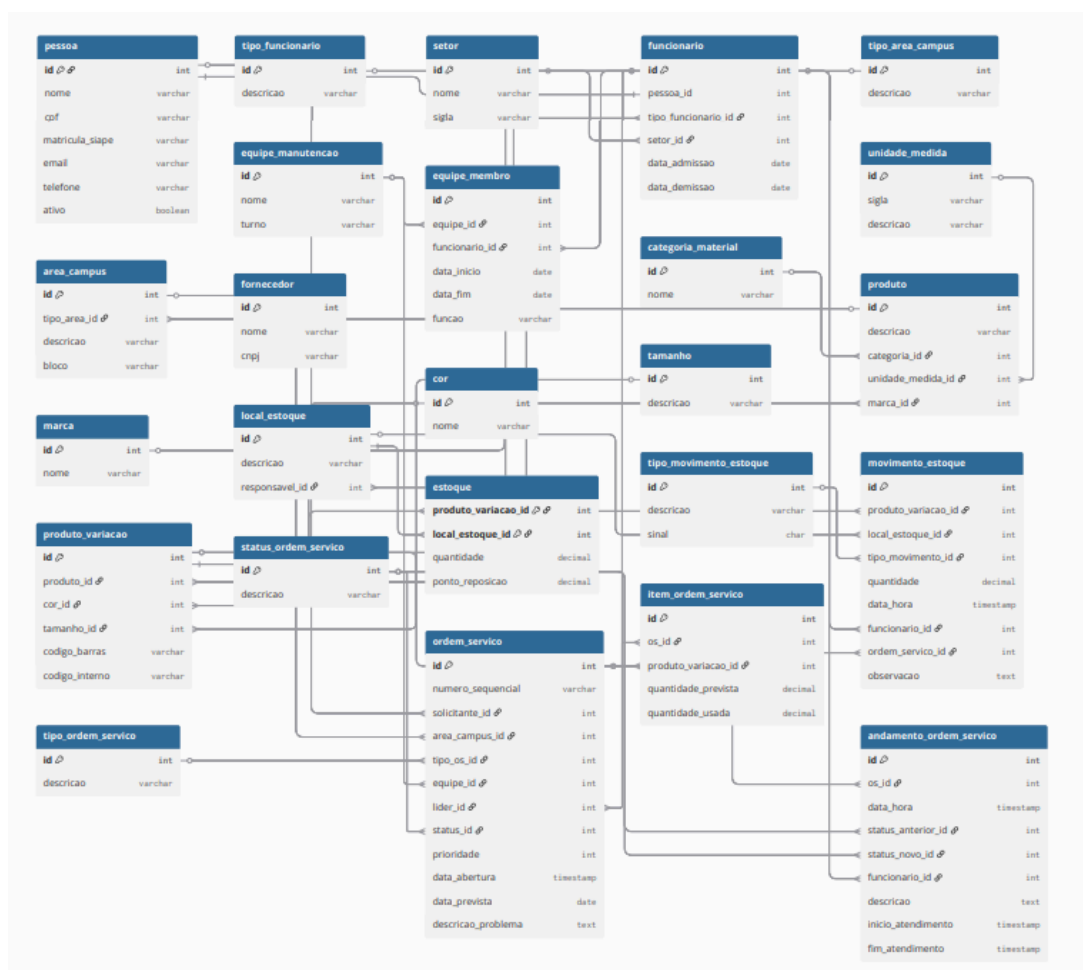
**MARACANAÚ, 2025**

## 1 - INTRODUÇÃO

Este projeto implementa o backend do sistema SIGEJ utilizando Java com Spring Boot. O banco de dados utilizado foi o PostgreSQL (via Docker), e a persistência foi feita com JDBC puro (sem ORM), conforme solicitado. O sistema atende aos 4 módulos principais: Gestão de RH (Equipes e Funcionários), Catálogo de Materiais, Controle de Estoque e Gestão de Ordens de Serviço (incluindo Infraestrutura do Campus).

## 2 - DIAGRAMA DER

A Figura abaixo apresenta o Diagrama Entidade-Relacionamento (DER) do sistema SIGEJ. O modelo é composto por 25 tabelas normalizadas, atendendo integralmente aos requisitos do projeto. As relações foram implementadas no banco de dados PostgreSQL através de chaves estrangeiras (Foreign Keys) e restrições de integridade (Constraints), garantindo a consistência dos dados sem o uso de ORM.



### 3 - RELATÓRIO DE EXECUÇÃO (testes no postman)

#### 3.1. Módulo de Recursos Humanos (RH)

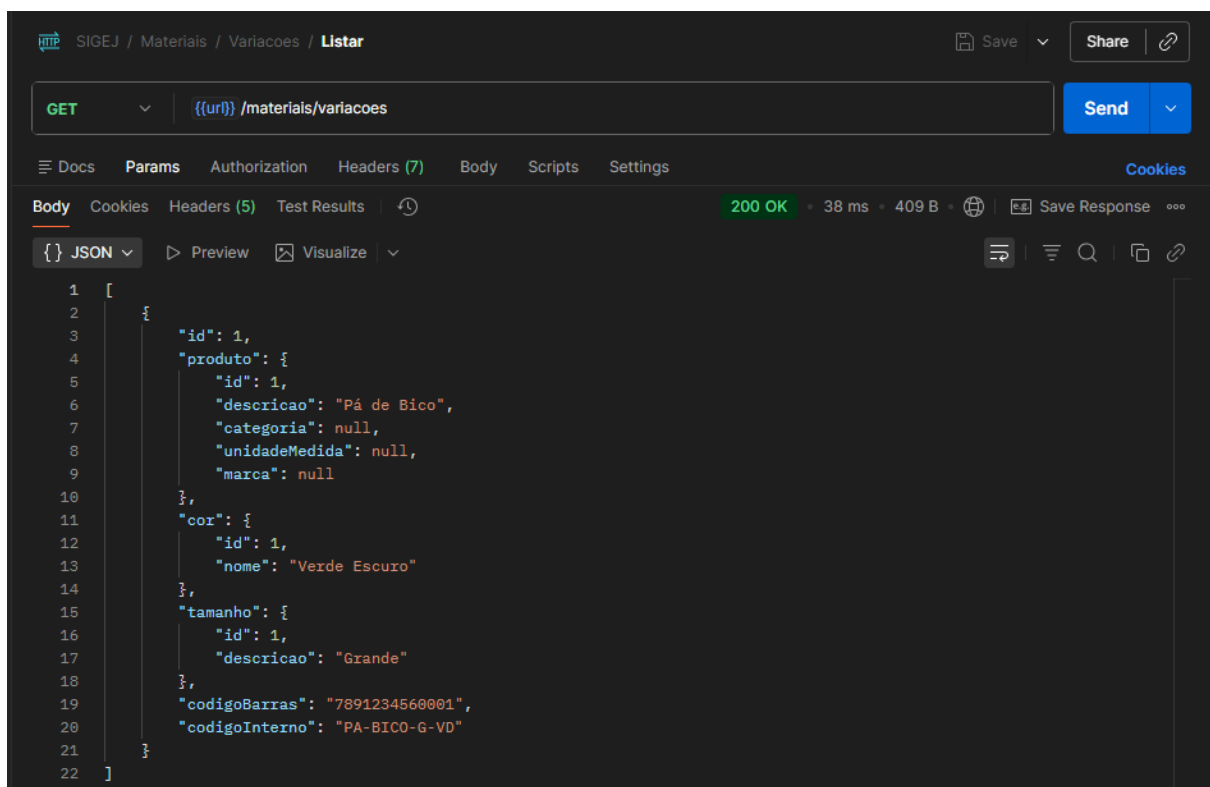
O retorno JSON abaixo exibe os dados do funcionário aninhados com as informações de sua Pessoa, Setor e Cargo, comprovando a integridade das relações.

The screenshot shows a Postman interface for a GET request to the endpoint `{{url}} /rh/funcionarios`. The response is a 200 OK status with a JSON body. The JSON structure is as follows:

```
[
  {
    "id": 1,
    "pessoa": {
      "id": 1,
      "nome": "Ivan da Silva",
      "cpf": null,
      "matriculaSiape": null,
      "email": null,
      "telefone": null,
      "ativo": null
    },
    "setor": {
      "id": 1,
      "nome": "Jardinagem e Paisagismo",
      "sigla": null
    },
    "tipo": {
      "id": 1,
      "descricao": "Tecnico Especializado"
    },
    "dataAdmissao": "2025-01-10",
    "dataDemissao": null
  }
]
```

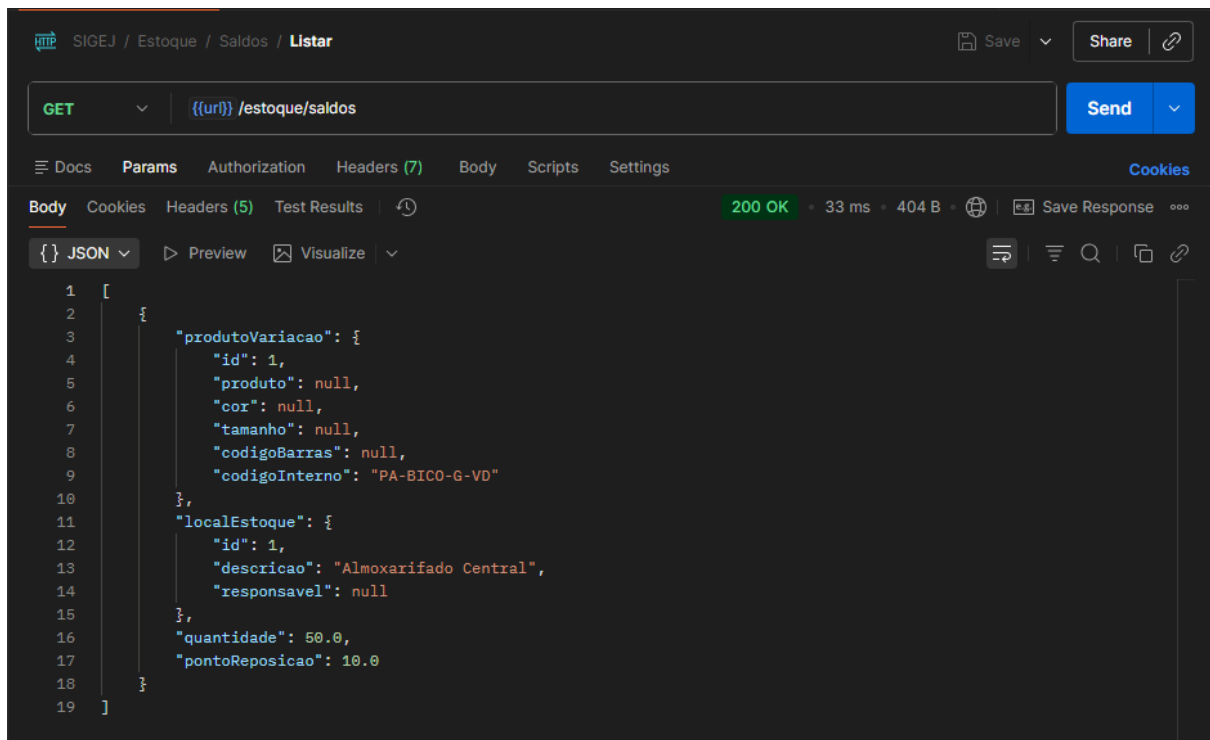
### 3.2. Módulo de Materiais e Produtos

A imagem abaixo mostra a recuperação de um Produto com suas variações específicas de Cor e Tamanho, essenciais para a gestão precisa do almoxarifado.



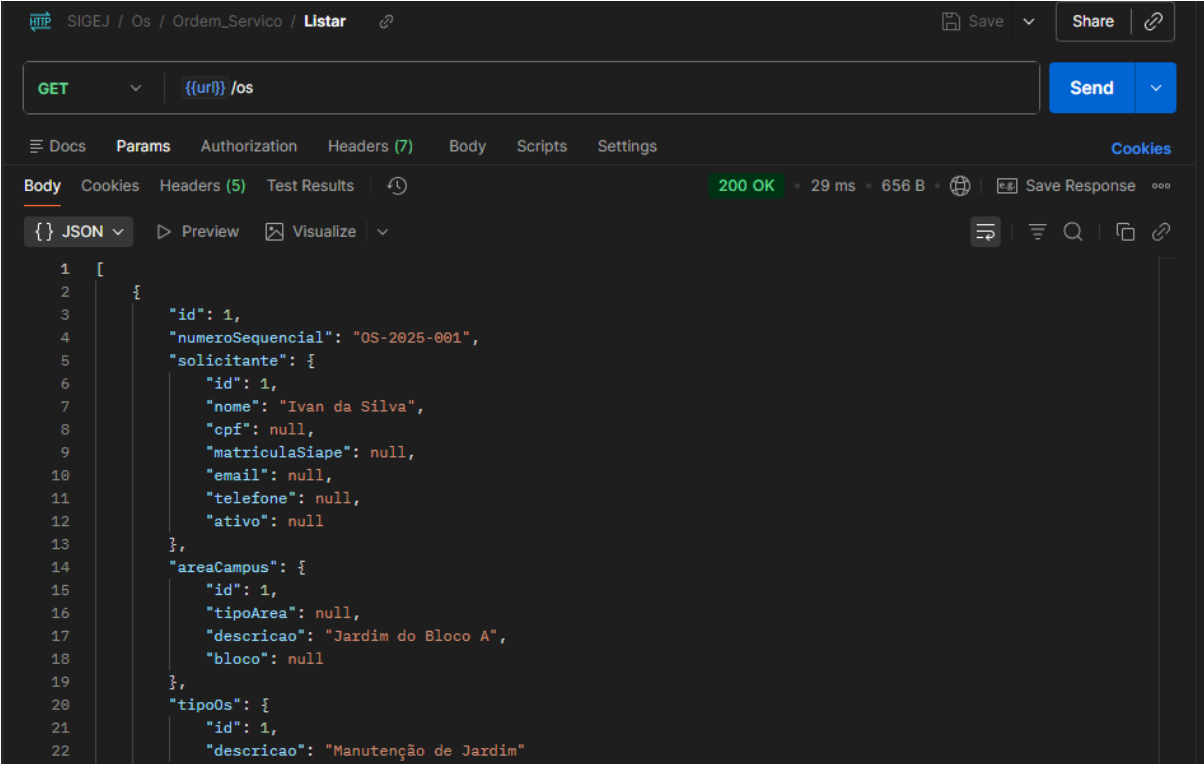
### 3.3. Controle de Estoque

O módulo de estoque controla o saldo atual de cada item por local de armazenamento. O sistema registra movimentações de entrada e saída e mantém o saldo atualizado, como demonstrado na consulta abaixo que exibe a quantidade disponível no 'Almoxarifado Central'.



### 3.4. Gestão de Ordens de Serviço (OS)

A Ordem de Serviço consolida as operações do sistema. Os prints abaixo comprovam a abertura de uma OS com prioridade definida, vinculada a um solicitante e a uma área do campus, pronta para execução pelas equipes de manutenção.



The screenshot shows a REST client interface with a GET request to `{{url}} /os`. The response is a 200 OK status with a 29 ms response time and 656 B of data. The response body is displayed in JSON format, showing a list of service orders. The first object in the list contains the following details:

```
1  [
2    {
3      "id": 1,
4      "numeroSequencial": "OS-2025-001",
5      "solicitante": {
6        "id": 1,
7        "nome": "Ivan da Silva",
8        "cpf": null,
9        "matriculaSiape": null,
10       "email": null,
11       "telefone": null,
12       "ativo": null
13     },
14     "areaCampus": {
15       "id": 1,
16       "tipoArea": null,
17       "descricao": "Jardim do Bloco A",
18       "bloco": null
19     },
20     "tipoOs": {
21       "id": 1,
22       "descricao": "Manutenção de Jardim"
```



The continuation of the JSON response shows the following details for the first service order:

```
23     },
24     "equipe": null,
25     "lider": null,
26     "status": {
27       "id": 2,
28       "descricao": "Em Andamento"
29     },
30     "prioridade": 5,
31     "dataAbertura": "2025-12-02T19:14:52.650091",
32     "dataPrevista": null,
33     "descricaoProblema": "Torneira vazando"
34   }
35 ]
```

The screenshot shows a REST client interface with a GET request to `{{url}} /os/1/andamentos`. The response is a 200 OK status with a 66 ms response time and 513 B of data. The response body is displayed in JSON format, showing a list of two items. The first item has an id of 1, os of null, dataHora of "2025-12-02T19:14:52.666891", statusAnterior with id 1 and descricao "Aberta", statusNovo with id 2 and descricao "Em Andamento", and funcionario with id 1, pessoa null, setor null, tipo null, dataAdmissao null, and dataDemissao null. The second item is partially visible at the bottom of the screenshot.

```
1 [
2   {
3     "id": 1,
4     "os": null,
5     "dataHora": "2025-12-02T19:14:52.666891",
6     "statusAnterior": {
7       "id": 1,
8       "descricao": "Aberta"
9     },
10    "statusNovo": {
11      "id": 2,
12      "descricao": "Em Andamento"
13    },
14    "funcionario": {
15      "id": 1,
16      "pessoa": null,
17      "setor": null,
18      "tipo": null,
19      "dataAdmissao": null,
20      "dataDemissao": null
21    },
22    "descricao": "Equipe iniciou o deslocamento",
23    "inicioAtendimento": null,
24    "fimAtendimento": null
25  }
26 ]
```

## 4 - CONCLUSÃO

O desenvolvimento do backend do SIGEJ atingiu todos os objetivos propostos. A aplicação foi construída em Java com Spring Boot, utilizando JDBC Template para manipulação direta de SQL, respeitando a restrição de não utilização de frameworks ORM.

O ambiente foi containerizado com Docker (PostgreSQL), garantindo portabilidade, e a automação do banco de dados foi implementada através de scripts SQL (schema.sql e data.sql) que garantem a recriação limpa e populada do ambiente a cada execução. O sistema atende aos 4 módulos principais (RH, Materiais, Estoque e OS), oferecendo uma API REST completa e testável.