

Instituto Federal de Educação, Ciência e Tecnologia do
Ceará
Campus Maracanaú

Victor Franklin Matias Silva

SIGEJ

Sistema de Gestão de Jardinagem e Manutenção

Disciplina: Banco de Dados
Curso: Ciência da Computação

Maracanaú - CE
29 de novembro de 2025

Sumário

1 Introdução.....	3
2 Tecnologias Utilizadas.....	3
3 Modelo de Dados.....	4
3.1 Diagrama Entidade-Relacionamento (DER).....	4
3.2 Principais Entidades.....	6
4 Funcionalidades Implementadas.....	6
4.1 Consultas Obrigatórias do Enunciado.....	7
5 Prints do Sistema em Uso.....	7

1 Introdução

O Sistema de Gestão de Jardinagem e Manutenção (SIGEJ), foi desenvolvido com o objetivo de organizar, automatizar e tornar mais eficiente o controle das atividades de manutenção realizadas no IFCE campus Maracanaú. A solução responde à necessidade de registrar ordens de serviço, gerenciar equipes, acompanhar o uso de materiais e visualizar o andamento das tarefas de forma estruturada.

Além de facilitar o fluxo das solicitações, o SIGEJ permite a rastreabilidade completa dos processos, desde a abertura de uma ordem de serviço até sua conclusão, passando pelo consumo de materiais, movimentações de estoque e histórico de atendimento. A implementação do sistema serve como apoio à gestão operacional do campus, promovendo maior transparência, redução de retrabalho e melhoria na tomada de decisões.

Este relatório apresenta a análise, modelagem, desenvolvimento e testes do sistema, incluindo a estrutura do banco de dados, consultas SQL obrigatórias, bem como instruções para execução do ambiente e evidências do funcionamento dos principais módulos.

2 Tecnologias Utilizadas

Para a implementação do SIGEJ foram utilizadas as seguintes tecnologias, bibliotecas e padrões de projeto:

- **Linguagem:** Python 3.10+, utilizando ambiente virtual próprio (.venv) gerenciado via Makefile.
- **Framework Web:** FastAPI, responsável por mapeamento das rotas REST em src/routes, organizadas por módulo (pessoa, funcionário, estoque, ordens de serviço, consultas, etc.).
- **Banco de Dados:** PostgreSQL, com todas as tabelas e restrições definidas manualmente em scripts/01_create_tables.sql. O arquivo docker-compose.yml sobe o banco de dados em um contêiner para facilitar desenvolvimento e testes.
- **Acesso a Dados (DAO):** A camada de acesso a dados está localizada em src/daos. Cada entidade possui um DAO responsável por executar comandos **INSERT**, **SELECT**, **UPDATE** e **DELETE** usando a biblioteca **psycopg2**.
- **Camada de Serviços:** Em src/services ficam as regras de negócio, que encapsulam e organizam operações mais complexas, combinando múltiplos DAOs.
- **Interfaces de Teste:** Como o frontend é opcional, as operações foram testadas diretamente via Swagger (FastAPI Docs), permitindo testar todos os endpoints de forma interativa sem necessidade de interface gráfica extra.
- **Seeds de Dados:** O arquivo scripts/02_seed.sql popula o banco com dados completos para testes: pessoas, funcionários, equipes, áreas do campus, produtos, estoque, OS e andamentos.
- **Ferramentas de Execução:** O projeto utiliza **Docker + Makefile** para simplificar o fluxo:

- **make up:** sobe o container PostgreSQL
- **make migrate:** aplica o script de criação do banco
- **make seed:** popula o banco
- **make main:** inicia a aplicação FastAPI
- **make down:** encerra os serviços

3 Modelo de Dados

O modelo de dados do SIGEJ foi projetado para representar de forma organizada as entidades e processos envolvidos na gestão de jardinagem e manutenção do campus. A estrutura segue o paradigma relacional, garantindo integridade referencial, consistência dos registros e clareza nas relações entre tabelas.

O banco é composto por entidades que representam pessoas, funcionários, equipes, áreas do campus, produtos, estoque e ordens de serviço, além de tabelas auxiliares para padronização de tipos e classificações. O modelo foi normalizado de forma a evitar redundâncias e permitir consultas eficientes, especialmente as exigidas no enunciado do projeto.

3.1 Diagrama Entidade-Relacionamento (DER)

O DER facilita a visualização da estrutura do banco e das dependências entre as tabelas. Abaixo está o diagrama que representa o modelo de dados implementado no projeto.

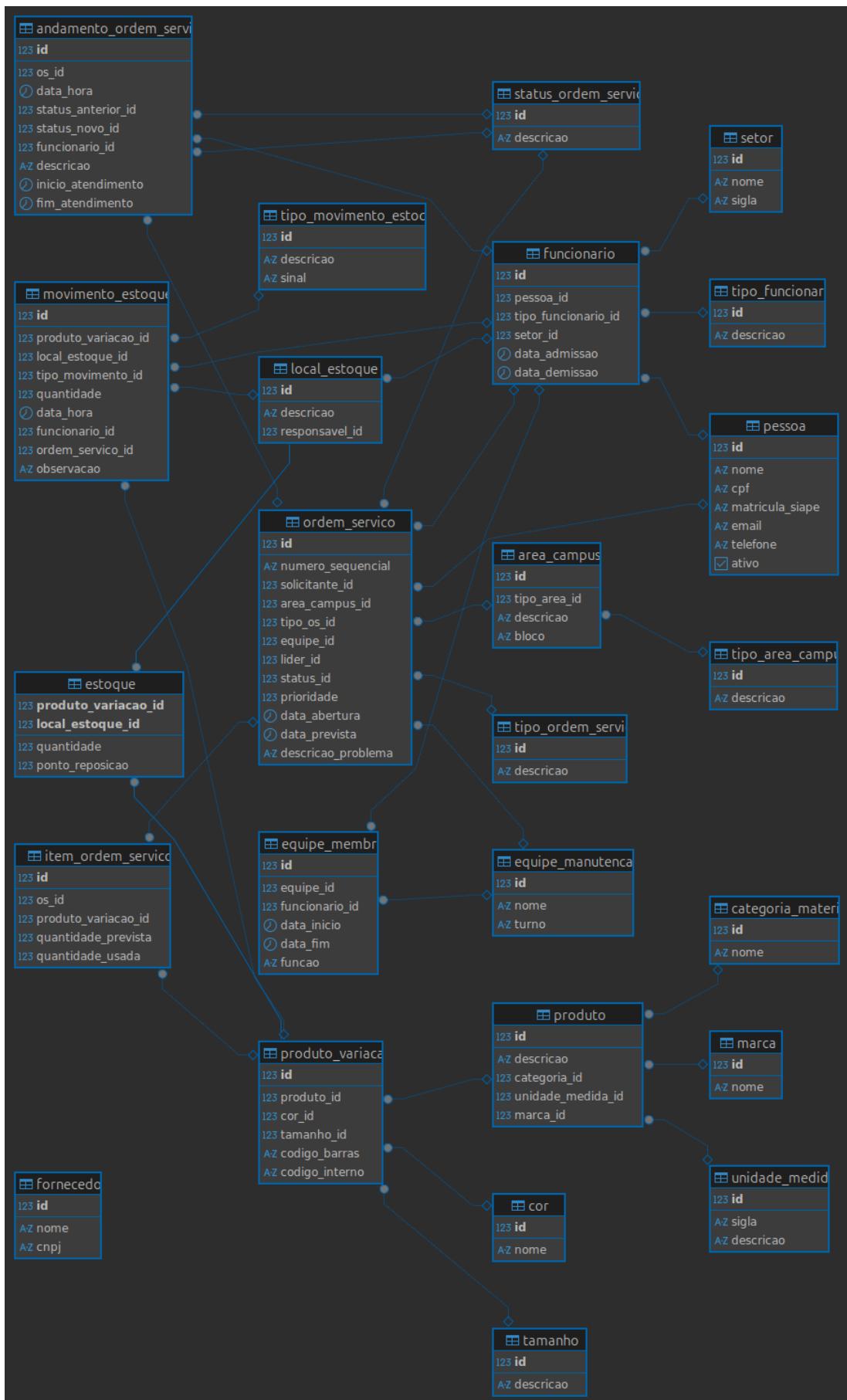


Figura 1: Diagrama Entidade-Relacionamento do SIGEJ

3.2 Principais Entidades

- **Pessoa:** Armazena dados pessoais (nome, CPF, e-mail, telefone).
- **Funcionário:** Especializa a entidade pessoa, vinculando-a a um tipo de funcionário e setor.
- **Equipe de Manutenção:** Representa as equipes responsáveis pelas atividades de campo, associadas a membros e turnos.
- **Área do Campus:** Locais onde as ordens de serviço são executadas.
- **Produto e Variações:** Representam os materiais e insumos utilizados nas manutenções.
- **Estoque:** Controla a quantidade disponível de cada produto em cada local.
- **Movimento de Estoque:** Registra entradas, saídas e consumo relacionado a ordens de serviço.
- **Ordem de Serviço (OS):** Entidade central do sistema, contendo prioridade, área, solicitante, equipe responsável e status.
- **Itens da OS:** Materiais previstos e consumidos na execução.
- **Andamento da OS:** Histórico detalhado de mudança de status e ações tomadas.

4 Funcionalidades Implementadas

Foram implementados CRUDs completos (Create, Read, Update, Delete) para as principais entidades do sistema, incluindo:

- **Pessoa**
- **Tipo de Funcionário**
- **Setor**
- **Funcionário**
- **Tipo de Área do Campus**
- **Área do Campus**
- **Equipe de Manutenção**
- **Membro de Equipe**
- **Categoria de Material**
- **Unidade de Medida**
- **Marca**
- **Produto**
- **Cor**
- **Tamanho**
- **Produto Variação**
- **Local de Estoque**
- **Estoque**
- **Tipo de Movimento de Estoque**
- **Tipo de Ordem de Serviço**
- **Status de Ordem de Serviço**
- **Ordem de Serviço**
- **Item da OS**
- **Andamento da OS**

Cada CRUD segue o mesmo padrão do sistema:

- **DAO**: manipulação direta no banco com SQL parametrizado
- **Service**: regras de negócio e validações
- **Routes (FastAPI)**: endpoints REST documentados automaticamente em /docs

4.1 Consultas Obrigatórias do Enunciado

Todas as consultas SQL exigidas no trabalho foram implementadas como **endpoints REST** na rota /consultas, com output em JSON:

1. **Ordens de serviço em aberto por prioridade e área**
2. **Materiais abaixo do ponto de reposição**
3. **Timeline (andamentos) de uma ordem de serviço**
4. **Consumo de materiais por equipe em determinado período**
5. **Quantidade de OS concluídas por tipo no ano informado**

5 Prints do Sistema em Uso

SIGEJ API 0.1.0 OAS 3.1

/openapi.json

Consultas

- GET** /api/v1/consultas/ordens-abertas Listar Ordens Abertas
- GET** /api/v1/consultas/materiais-criticos Listar Materiais Criticos
- GET** /api/v1/consultas/timeline/{os_id} Listar Timeline Os
- GET** /api/v1/consultas/consumo-equipe Listar Consumo Por Equipe
- GET** /api/v1/consultas/os-concluidas/{ano} Listar Os Concluidas Por Tipo

Figura 2: Endpoint de consultas

The screenshot shows the API documentation for the 'ordens-abertas' endpoint. It includes a status code of 200, a response body example, and response headers.

Response body

```
[{"id": 3, "numero_sequencial": "OS-2025-0003", "prioridade": 1, "area": "Praca Central", "tipo_servico": "Manutenção", "solicitante": "Carla Santos", "data_abertura": "2025-10-10T10:00:00"}, {"id": 1, "numero_sequencial": "OS-2025-0001", "prioridade": 2, "area": "Bloco 3 - Jardins", "tipo_servico": "Manutenção", "solicitante": "Ana Pereira", "data_abertura": "2025-11-01T09:00:00"}, {"id": 2, "numero_sequencial": "OS-2025-0002", "prioridade": 3, "area": "Bloco 3 - Estacionamento", "tipo_servico": "Limpesa", "solicitante": "Bruno Oliveira", "data_abertura": "2025-11-01T09:00:00"}]
```

Response headers

```
content-length: 550
content-type: application/json
date: Sun, 30 Nov 2025 00:48:46 GMT
server: unicorn
```

Figura 3: Consulta de ordens abertas

SIGEJ API 0.1.0 OAS 3.1
[/openapi.json](#)

- Consultas
 - Pessoas
 - Tipo Funcionario
 - Funcionario
 - Setor
 - Tipo Area Campus
 - Area Campus
 - Equipe Manutencao
 - Equipe Membro
 - Categoria Material
 - Unidade Medida
 - Fornecedor
 - Marca
 - Cor
 - Tamanho
 - Produto
 - Produto Variacao
 - Local Estoque
 - Tipo Movimento Estoque
 - Movimento Estoque
 - Estoque
 - Tipo Ordem Servico
 - Status Ordem Servico
 - Ordem Servico
 - Item Ordem Servico
 - Andamento Ordem Servico
- Schemas

Figura 4: Rotas de todas as entidades do banco

SIGEJ API 0.1.0 OAS 3.1
[/openapi.json](#)

- Consultas
- Pessoas
 - GET /api/v1/pessoas/ Listar Pessoas
 - POST /api/v1/pessoas/ Criar Pessoa
 - GET /api/v1/pessoas/{pessoa_id} Obter Pessoa
 - PUT /api/v1/pessoas/{pessoa_id} Atualizar Pessoa
 - DELETE /api/v1/pessoas/{pessoa_id} Remover Pessoa

Figura 5: Crud de pessoas