Detalhamento do Projeto Final da Disciplina

Bruno Eduardo -, -
Matheus de -, -
Gustavo Alencar -, -

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
Banco de dados
14 de julho de 2025

-@aluno.unb.br, -aluno.unb.br, -@aluno.unb.br

Resumo. *O presente relatório detalha o desenvolvimento e a funcionalidade do Sistema de Controle de Qualidade de Patrimônio (SCQP), uma ferramenta estratégica projetada para otimizar a gestão de bens e infraestrutura de uma organização. O objetivo principal do SCQP é proporcionar uma visão abrangente e atualizada do patrimônio, integrando informações detalhadas sobre espaços físicos, como edifícios e salas, e os equipamentos neles contidos.*

Sumário

1. Introdução

Este projeto apresenta o **Sistema de Controle de Qualidade de Patrimônio (SCQP)**, desenvolvido com o objetivo de otimizar a gestão dos bens patrimoniais da organização. O SCQP integra informações detalhadas sobre espaços físicos (salas e edifícios) e seus respectivos equipamentos, proporcionando uma visão abrangente e atualizada do patrimônio.

Adicionalmente, o sistema monitora o status operacional desses ativos, identificando equipamentos em funcionamento ou com avarias, o que possibilita a implementação de manutenção preventiva e a redução de interrupções operacionais. Uma funcionalidade essencial do SCQP é o registro estruturado de ocorrências, permitindo que usuários reportem problemas, os quais são detalhadamente documentados para otimizar o fluxo de trabalho das equipes de manutenção.

As tabelas estão organizadas em três grandes grupos:

- **Estrutura Organizacional:** Representa a infraestrutura acadêmica e administrativa da instituição, abrangendo os prédios físicos (`Predio`), departamentos vinculados a esses prédios (`Departamento`) e os cursos oferecidos (`Curso`).
- **Gestão de Usuários:** Controla as informações dos usuários do sistema (`Usuario`), distinguindo entre funcionários (`Funcionario`), docentes com vínculo departamental (`Docente`) e discentes vinculados a cursos específicos (`Discente`).
- **Infraestrutura Física e Suporte Técnico:** Abrange o gerenciamento de salas físicas (`Sala`) e equipamentos instalados (`Equipamento`), classificados por tipo (`TipoEquipamento`) e status operacional (`StatusEquipamento`). Também contempla o registro de ocorrências técnicas (`Ocorrencia`) e os serviços de manutenção realizados nos equipamentos (`Manutencao`), com rastreabilidade do responsável e do problema relatado.

Todo o pojeto esta no github: https://github.com/brnduol/SCQP

2. Diagrama de Entidade–Relacionamento

O sistema proposto é composto por 13 entidades interconectadas, cuidadosamente definidas conforme o escopo do projeto. Essas entidades representam diferentes áreas funcionais do sistema e foram organizadas em três grupos principais, conforme mostrado na Tabela 1.

Tabela 1. Entidades do Banco de Dados

Grupo	Entidades
Estrutura Organizacional	`Predio`, `Departamento`, `Curso`
Gestão de Usuários	`Usuario`, `Funcionario`, `Docente`, `Discente`
Infraestrutura e Suporte Técnico	`Sala`, `TipoEquipamento`, `StatusEquipamento`, `Equipamento`, `Ocorrencia`, `Manutencao`

O **diagrama entidade–relacionamento (ER)** do sistema foi elaborado com base na modelagem conceitual das entidades e em suas respectivas interações. Durante seu

desenvolvimento, o modelo passou por refinamentos e validações colaborativas entre os integrantes da equipe, assegurando sua coerência, completude e aderência às necessidades do projeto.

A partir dessa análise e estruturação, foi desenvolvido o diagrama ER definitivo, utilizando a ferramenta **BRModelo**, conforme apresentado na Figura 1.

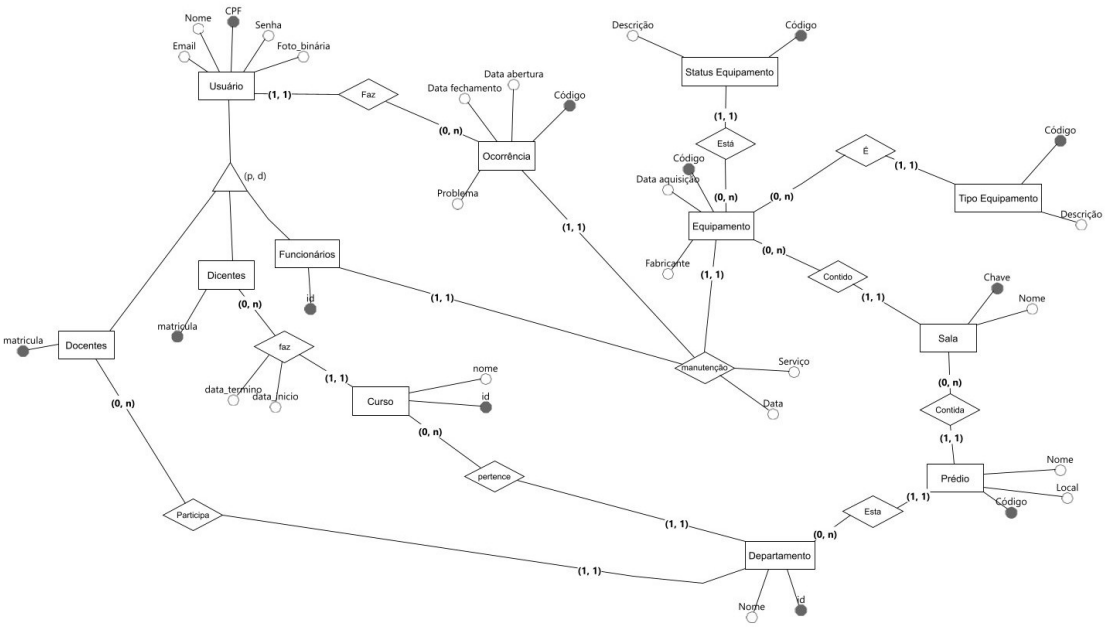

Figura 1. Diagrama entidade–relacionamento desenvolvido no BRModelo

3. Modelo Relacional

Dando continuidade ao desenvolvimento do sistema, o próximo passo foi a conversão do diagrama entidade–relacionamento (ER) para o **modelo relacional lógico**. Essa etapa tem como objetivo representar, em termos de tabelas e chaves, a estrutura conceitual do banco de dados em um formato que possa ser implementado diretamente em um SGBD relacional.

Utilizamos a ferramenta **BRModelo** para realizar essa conversão de forma automatizada. No entanto, embora a geração automática do modelo relacional seja prática, ela nem sempre é perfeita. Durante a conversão, foi necessário realizar ajustes manuais para garantir que a estrutura estivesse de acordo com os requisitos do sistema.

Um exemplo dessa limitação é a representação da *generalização/especialização* presente no modelo conceitual. O BRModelo não trata corretamente esse recurso, resultando na criação de uma única tabela contendo todos os atributos das entidades especializadas, o que pode não refletir a estrutura ideal em certos casos.

Após os devidos ajustes, obteve-se o modelo relacional representado na Figura 2.

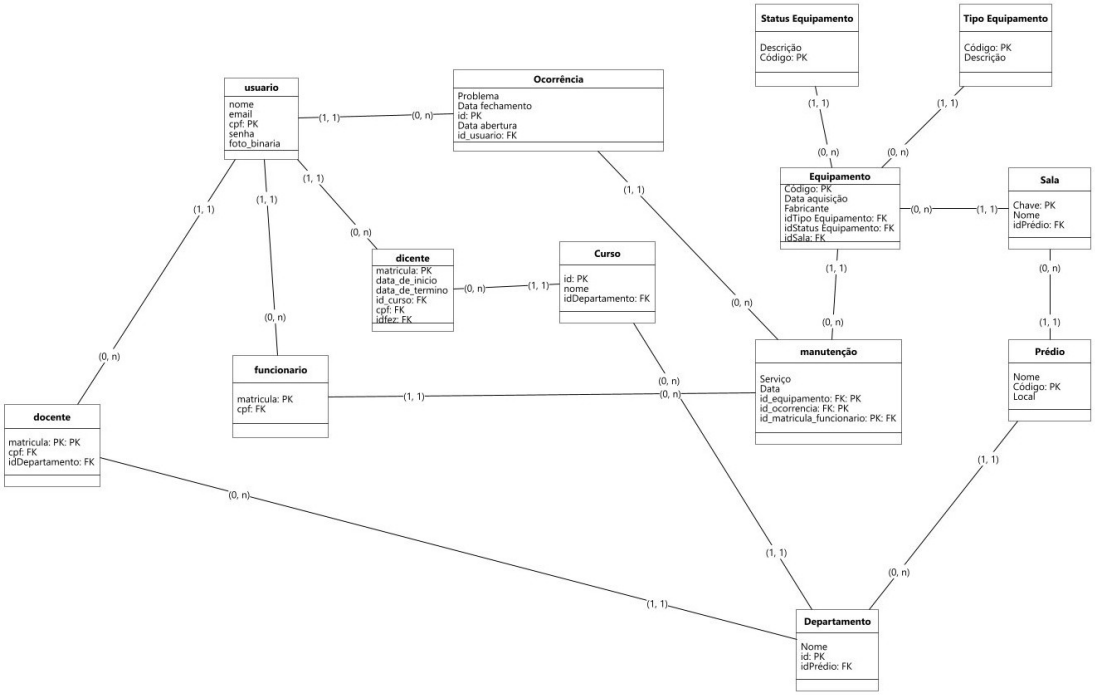

Figura 2. Modelo relacional gerado e ajustado no BRModelo

4. Álgebra relacional

Nesta seção, apresentamos cinco consultas em álgebra relacional, cada uma envolvendo pelo menos três tabelas do banco de dados desenvolvido. As expressões são seguidas de explicações para facilitar a compreensão.

4.1. Consulta 1: Discentes, seus cursos e departamentos

$$\pi_{\text{Usuario.nome, Curso.nome, Departamento.nome}}(Usuario \bowtie_{\text{Usuario.cpf = Discente.cpf}} (Discente \bowtie_{\text{Discente.id\_curso = Curso.id}} (Curso \bowtie_{\text{Curso.id\_departamento = Departamento.id}} Departamento)))$$

A consulta retorna os nomes dos discentes, seus cursos e os departamentos correspondentes. Envolve as tabelas `Usuario`, `Discente`, `Curso` e `Departamento`.

4.2. Consulta 2: Ocorrências com dados do usuário, equipamento e sala

$$\pi_{\text{Ocorrencia.id, Usuario.nome, Equipamento.id, Sala.nome}}(Ocorrencia \bowtie_{\text{Ocorrencia.id\_usuario = Usuario.cpf}} (Manutencao \bowtie_{\text{Manutencao.id\_ocorrencia = Ocorrencia.id}} (Equipamento \bowtie_{\text{Equipamento.id = Manutencao.id\_equipamento}} (Sala \bowtie_{\text{Sala.id = Equipamento.id\_sala}} Sala))))$$

Lista informações sobre as ocorrências, o usuário que registrou, o equipamento envolvido e a sala onde este está instalado.

4.3. Consulta 3: Docentes, departamentos e prédios alocados

$$\pi_{\text{Usuario.nome, Departamento.nome, Predio.nome}}(\\ Usuario \bowtie_{\text{Usuario.cpf = Docente.cpf}} (\\ Docente \bowtie_{\text{Docente.id\_departamento = Departamento.id}} (\\ Departamento \bowtie_{\text{Departamento.id\_predio = Predio.id}} Predio)))$$

Exibe os nomes dos docentes, os departamentos aos quais pertencem e os prédios onde esses departamentos estão localizados.

4.4. Consulta 4: Equipamentos com tipo, status, sala e prédio

$$\pi_{\text{Equipamento.id, TipoEquipamento.descricao, StatusEquipamento.descricao, Sala.nome, Predio.nome}}(\\ Equipamento \bowtie_{\text{Equipamento.id\_tipo = TipoEquipamento.id}} (\\ Equipamento \bowtie_{\text{Equipamento.id\_status = StatusEquipamento.id}} (\\ Sala \bowtie_{\text{Sala.id = Equipamento.id\_sala}} (\\ Predio \bowtie_{\text{Predio.id = Sala.id\_predio}} Predio))))$$

Apresenta os equipamentos cadastrados com seus respectivos tipos, status atuais, as salas e prédios onde estão instalados.

4.5. Consulta 5: Manutenções com funcionário, equipamento e ocorrência

$$\pi_{\text{Usuario.nome, Equipamento.id, Ocorrencia.problema, Manutencao.servico}}(\\ Manutencao \bowtie_{\text{Manutencao.id\_funcionario = Funcionario.matricula}} (\\ Funcionario \bowtie_{\text{Funcionario.cpf = Usuario.cpf}} (\\ Manutencao \bowtie_{\text{Manutencao.id\_equipamento = Equipamento.id}} (\\ Ocorrencia \bowtie_{\text{Ocorrencia.id = Manutencao.id\_ocorrencia}} Ocorrencia))))$$

Retorna os serviços de manutenção realizados, incluindo o nome do funcionário responsável, o equipamento atendido e o problema relatado na ocorrência.

5. Formas Normais

Foi solicitado que o sistema esteja em conformidade com a **Terceira Forma Normal (3FN)**. Durante o desenvolvimento dos diagramas e da modelagem do banco de dados, tomamos o cuidado de aplicar as regras de normalização desde o início, garantindo a consistência e integridade dos dados.

Contudo, para demonstrar na prática o processo de normalização, realizaremos a **desnormalização** de algumas tabelas, simulando um cenário inicial não normalizado. A partir disso, aplicaremos sucessivamente as formas normais — 1FN, 2FN e 3FN — para evidenciar a transformação da estrutura até alcançar um modelo ideal.

Utilizaremos como base as seguintes entidades reais do sistema:

- `Discente` (aluno)
- `Curso`
- `Departamento`

Assim, os dados são inicialmente representados por uma única tabela não normalizada:

Tabela 2. Tabela Não Normalizada: Dados de Alunos e Cursos

CPF	Nome do Aluno	Curso	Data de Início	Departamento	Prédio	Local
11111111111	Ana Souza	Engenharia	2022-01-10	Exatas	Prédio A	Bloco Central
22222222222	João Lima	Engenharia	2022-01-10	Exatas	Prédio A	Bloco Central
33333333333	Carla Dias	História	2023-02-15	Humanas	Prédio B	Ala Norte

5.1. Primeira Forma Normal (1FN)

A **Primeira Forma Normal (1FN)** estabelece que todas as colunas de uma tabela devem conter apenas valores **atômicos**, ou seja, indivisíveis. Cada célula deve conter um único valor, e não listas ou conjuntos de dados.

Ao analisar a Tabela 2, observamos que, embora os atributos estejam atomicamente definidos, ainda há presença de **redundância de dados**. Por exemplo, os campos `Departamento`, `Prédio` e `Local` são repetidos em diversas linhas, o que pode acarretar anomalias em atualizações e desperdício de espaço.

A seguir, mantemos a estrutura com atributos atômicos, caracterizando a Tabela em conformidade com a 1FN, mas ainda sem eliminação das redundâncias:

Tabela 3. Tabela na Primeira Forma Normal (1FN)

CPF	Aluno	Curso	Data de Início	Departamento	Prédio	Local
11111111111	Ana Souza	Engenharia	2022-01-10	Exatas	Prédio A	Bloco Central
22222222222	João Lima	Engenharia	2022-01-10	Exatas	Prédio A	Bloco Central
33333333333	Carla Dias	História	2023-02-15	Humanas	Prédio B	Ala Norte

5.2. Segunda Forma Normal (2FN)

A **Segunda Forma Normal (2FN)** exige que a tabela esteja, antes de tudo, em conformidade com a 1FN, e que todos os atributos não-chave sejam **totalmente dependentes da chave primária**. Isso significa que **não pode haver dependência parcial** — ou seja, nenhum atributo deve depender apenas de parte da chave primária composta.

Na Tabela 2, se considerássemos como chave primária a combinação `CPF + Curso`, atributos como `Departamento`, `Prédio` e `Local` dependeriam apenas do curso, e não do aluno específico. Essa é uma violação da 2FN.

Para atender à 2FN, foi necessário separar os dados em duas tabelas: uma com as informações específicas dos discentes (alunos) e outra com os dados dos cursos. Com isso, eliminamos as dependências parciais.

Tabela 4. Discente (2FN)

CPF	Aluno	Data de Início	ID Curso
11111111111	Ana Souza	2022-01-10	1
22222222222	João Lima	2022-01-10	1
33333333333	Carla Dias	2023-02-15	2

Tabela 5. Curso (2FN)

ID Curso	Nome do Curso	Departamento
1	Engenharia	Exatas
2	História	Humanas

5.3. Terceira Forma Normal (3FN)

A **Terceira Forma Normal (3FN)** exige que a tabela esteja, no mínimo, em conformidade com a 2FN e que todos os atributos não-chave sejam **diretamente dependentes da chave primária**. Isso significa que devem ser eliminadas as **dependências transitivas**, nas quais um atributo depende de outro que não é chave, e este por sua vez depende da chave primária.

No exemplo anterior, temos as seguintes dependências transitivas:

- O `Curso` depende do `Departamento`
- O `Departamento` depende do `Prédio`

Essas relações intermediárias devem ser extraídas e modeladas em tabelas separadas, de forma que cada entidade represente uma única temática, evitando anomalias de atualização e facilitando a manutenção dos dados.

A seguir, apresentamos a estrutura final em conformidade com a 3FN:

Tabela 6. Discente (3FN)

Matrícula	CPF	Data de Início	Data de Término	ID Curso
1001	11111111111	2022-01-10	–	1
1002	22222222222	2022-01-10	–	1
1003	33333333333	2023-02-15	–	2

Tabela 7. Curso (3FN)

ID Curso	Nome	ID Departamento
1	Engenharia	10
2	História	20

Tabela 8. Departamento (3FN)

ID Departamento	Nome	ID Prédio
10	Exatas	100
20	Humanas	200

6. Diagrama da Camada de Mapeamento

Nesta seção é apresentada a **camada de mapeamento objeto-relacional (ORM)** do sistema, a qual estabelece a correspondência entre as tabelas do banco de dados relacional e as classes definidas em Python.

Foi utilizada a biblioteca **SQLAlchemy**, amplamente adotada para construção de sistemas baseados em ORM com Python. Ela permite que cada tabela do banco seja representada por uma classe, e que cada coluna e chave estrangeira sejam modeladas como atributos ou relacionamentos, facilitando a manipulação dos dados de forma orientada a objetos.

A seguir, apresenta-se o diagrama visual da camada de mapeamento, que reflete a estrutura lógica do banco de dados implementado:

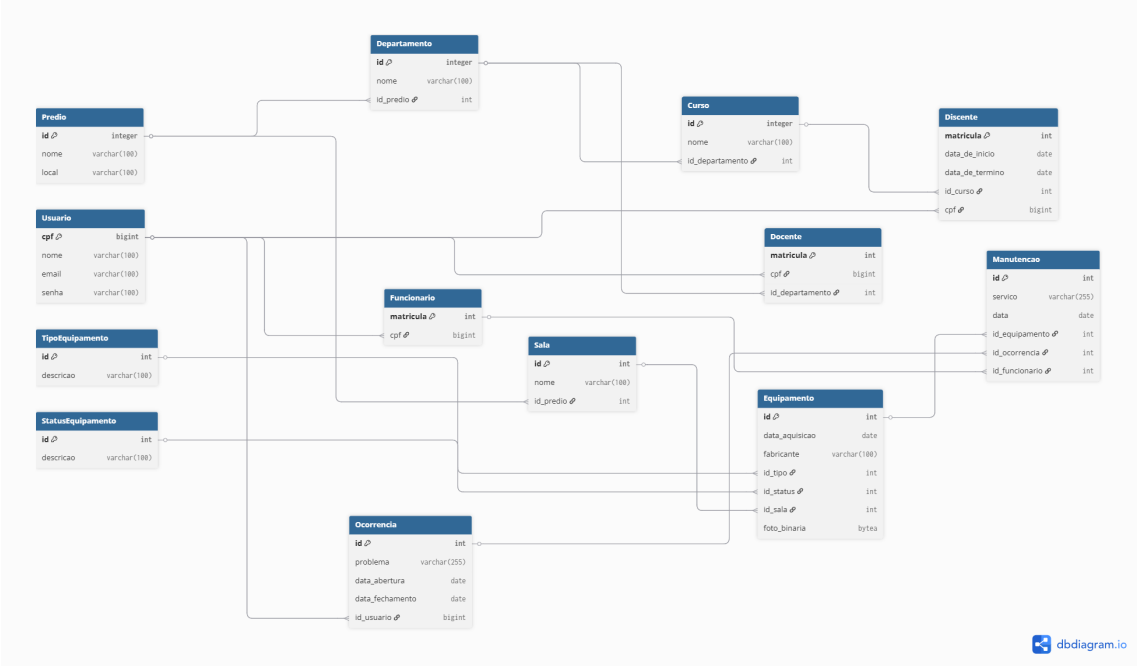

Figura 3. Diagrama da camada de mapeamento objeto-relacional

O diagrama ilustra como as classes `Usuario`, `Discente`, `Curso`, `Departamento`, entre outras, estão interligadas, respeitando os relacionamentos definidos no modelo lógico.

Além disso, recursos como `foreign keys` e `relationships` do SQLAlchemy foram utilizados para garantir integridade referencial, facilitar a navegação entre entidades e permitir operações encadeadas entre os objetos no código da aplicação.

Para isso, cada tabela foi convertida em uma classe Python, como demonstrado a seguir:

Listing 1. Exemplo de mapeamento ORM com SQLAlchemy

```
class Predio(db.Model):
    __tablename__ = 'Predio'
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
```

```
    nome = db.Column(db.String(100))
    local = db.Column(db.String(100))
    departamentos = db.relationship('Departamento', backref='predio',
    cascade='all, delete')
    salas = db.relationship('Sala', backref='predio', cascade='all,
    delete')


class Departamento(db.Model):
    __tablename__ = 'Departamento'
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    nome = db.Column(db.String(100))
    id_predio = db.Column(db.Integer, db.ForeignKey('Predio.id',
    ondelete='CASCADE'))
    cursos = db.relationship('Curso', backref='departamento', cascade='
    all, delete')
    docentes = db.relationship('Docente', backref='departamento',
    cascade='all, delete')

 # Resto do codigo ocultado
```

7. Views e Procedures

No desenvolvimento do nosso banco de dados, implementamos algumas *views* e *procedures*. Inicialmente, essas funcionalidades foram incorporadas diretamente na lógica da aplicação utilizando `SQLAlchemy`. No entanto, também desenvolvemos versoes de algumas em SQL puro, que são apresentadas a seguir para fins de documentação.

7.1. Procedure para exclusão de ocorrência por ID

A procedure abaixo realiza a exclusão de uma ocorrência com base no seu ID. Caso o ID informado não seja encontrado, uma mensagem é exibida indicando que nenhuma ocorrência foi excluída.

Listing 2. Procedure para excluir ocorrência por ID

```
CREATE OR REPLACE PROCEDURE excluir_ocorrencia_por_id(id_oc INTEGER)
LANGUAGE plpgsql
AS $$
BEGIN
    DELETE FROM ocorrencia WHERE id = id_oc;

    IF NOT FOUND THEN
        RAISE NOTICE 'Ocorrencia com ID % nao encontrada.', id_oc;
    ELSE
        RAISE NOTICE 'Ocorrencia com ID % excluida com sucesso.', id_oc
    ;
    END IF;
END;
$$;
```

7.2. View de ocorrências com informações dos usuários

A view a seguir foi criada para facilitar o acesso às informações das ocorrências juntamente com os dados dos usuários relacionados, permitindo consultas mais práticas e legíveis.

Listing 3. View de ocorrências com usuários

```sql
CREATE OR REPLACE VIEW ocorrencias_com_usuarios AS
SELECT
    o.id,
    o.problema,
    o.data_abertura,
    o.data_fechamento,
    u.nome AS nome_usuario
FROM
    ocorrencia o
JOIN
    usuario u ON o.id_usuario = u.cpf;
```