Grupo:

Pedro Rafael Santos Pignata - **RA**:24101218

Pedro Pereira Dutra - **RA:** 24101026

Yuri Buenos Aires Santana - **RA**: 24101289

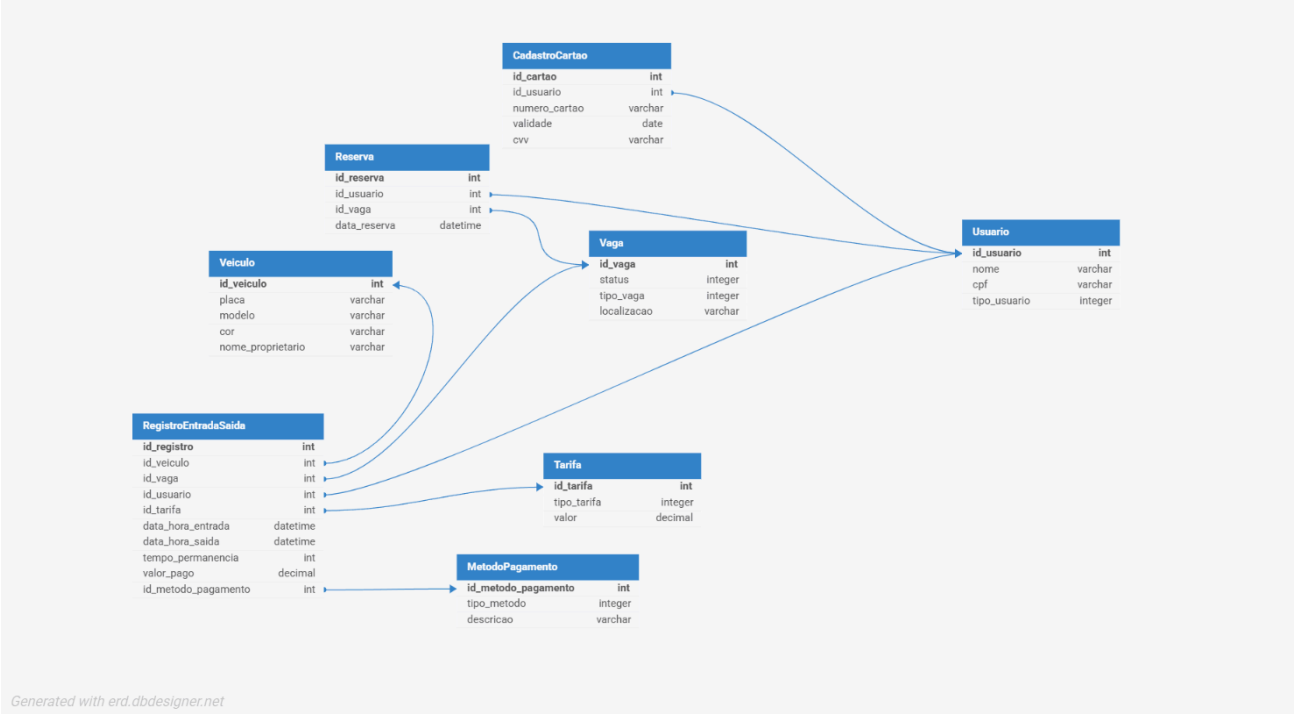
Rafael Lopes Dias - **RA:** 24101237

Renato Portilho Costa - **RA:** 24101301

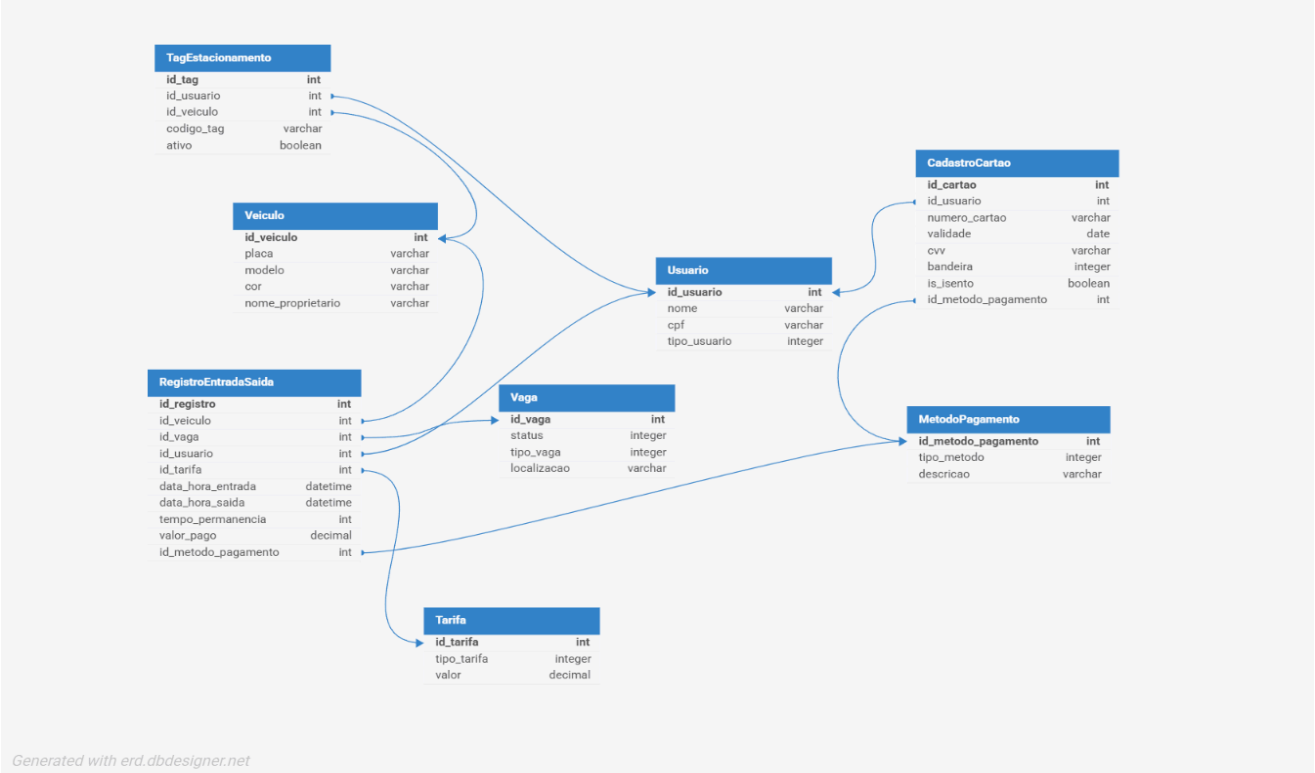
Gabriel Delforge - **RA:** 24101263

Diagramas de Entidade-Relacionamento

Antigo:



Atual:



Laboratório Respostas

1.

Seleção (σ) e Projeção (π):

πid_professor,tag(σstatus='Ativa'(TAG_ESTACIONAMENTO))\pi_{id_professor, tag}(\sigma_{status = 'Ativa'}(TAG_ESTACIONAMENTO))

Isso filtra apenas as tags ativas e retorna os campos id_professor e tag.

2.

Diferença de Conjuntos (-):

πid_professor(RESERVAS)−πid_professor(TAG_ESTACIONAMENTO)\pi_{id_professor}(RESERVAS) - \pi_{id_professor}(TAG_ESTACIONAMENTO)

Isso retorna professores que estavam na tabela RESERVAS mas não aparecem na tabela TAG_ESTACIONAMENTO.

3.

Junção (⋈) e Seleção (σ):

TAG_ESTACIONAMENTO⋈TAG_ESTACIONAMENTO.id_professor=PAGAMENTO.id_professorσisencao_visa='Sim'(PAGAMENTO)TAG_ESTACIONAMENTO \bowtie_{TAG_ESTACIONAMENTO.id_professor = PAGAMENTO.id_professor} \sigma_{isencao_visa = 'Sim'}(PAGAMENTO)

Isso combina os professores da tabela TAG_ESTACIONAMENTO com os pagamentos e filtra aqueles com isencao_visa = 'Sim'.

4.

Seleção (σ):

σstatus='Inativa'(TAG_ESTACIONAMENTO)\sigma_{status = 'Inativa'}(TAG_ESTACIONAMENTO)

Isso retorna professores com status = 'Inativa'.

5.

Seleção (σ):

σmetodo_pagamento='Visa'(PAGAMENTO)\sigma_{metodo_pagamento = 'Visa'}(PAGAMENTO)

Isso filtra apenas pagamentos feitos com Visa.

6.

Junção (⋈) e Seleção (σ):

σstatus='Ativa'(TAG_ESTACIONAMENTO)⋈TAG_ESTACIONAMENTO.id_professor=PAGAMENTO.id_professorPAGAMENTO\sigma_{status = 'Ativa'}(TAG_ESTACIONAMENTO) \bowtie_{TAG_ESTACIONAMENTO.id_professor = PAGAMENTO.id_professor} PAGAMENTO

Isso retorna professores com tag ativa e pagamento cadastrado.

7.

Diferença de Conjuntos (-):

πid_professor(DOCENTE)−πid_professor(TAG_ESTACIONAMENTO)\pi_{id_professor}(DOCENTE) - \pi_{id_professor}(TAG_ESTACIONAMENTO)

Isso retorna os professores que ainda não possuem tag.

8.

Projeção (π):

πid_professor,tag(TAG_ESTACIONAMENTO)\pi_{id_professor, tag}(TAG_ESTACIONAMENTO)

Isso remove colunas desnecessárias e mantém **apenas** os campos **id_professor** e **tag**.

9.

Diferença de Conjuntos (-):

πid_professor(DOCENTE)−πid_professor(PAGAMENTO)\pi_{id_professor}(DOCENTE) - \pi_{id_professor}(PAGAMENTO)

Isso retorna os professores cadastrados, mas sem pagamento registrado.

10.

Junção (⋈) e Seleção (σ):

σstatus='Ativa'(TAG_ESTACIONAMENTO)⋈TAG_ESTACIONAMENTO.id_professor=PROBLEMASTECNICOS.id_professorPROBLEMASTECNICOS\sigma_{status = 'Ativa'}(TAG_ESTACIONAMENTO) \bowtie_{TAG_ESTACIONAMENTO.id_professor = PROBLEMAS_TECNICOS.id_professor} PROBLEMAS_TECNICOS

Isso retorna professores com tag ativa que possuem problema técnico cadastrado.

Relatório Técnico:

1. Introdução

Este relatório tem como objetivo documentar o processo de modelagem do sistema de controle de estacionamento para os docentes do IDP, bem como as alterações

implementadas devido ao acordo firmado com a empresa de tags de estacionamento e com a bandeira Visa.

---

2. Modelagem do Sistema

O sistema foi inicialmente projetado para gerenciar reservas de vagas para os docentes do IDP. A modelagem seguiu os padrões de banco de dados relacionais, estruturando as seguintes entidades principais:

- **Usuario**: representa os usuarios que utilizam o estacionamento.
- **Reserva**: registro das reservas feitas para as vagas.
- **Pagamento**: controle dos pagamentos das mensalidades do estacionamento.

2.1. Modelo Inicial

O modelo inicial possuía uma tabela de **reservas**, onde os docentes cadastravam suas solicitações diárias para o estacionamento. A verificação de disponibilidade era feita manualmente, o que gerava filas e atrasos no acesso ao campus.

---

3. Atualizações Implementadas

Com a implementação do novo sistema de tags, houve alterações significativas na modelagem do banco de dados e na gestão do estacionamento.

3.1. Remoção da Tabela de Reservas

A necessidade de agendamentos manuais foi eliminada, tornando a tabela de **reservas** obsoleta. Em seu lugar, foi criada a tabela **tag_estacionamento**.

3.2. Inclusão da Tabela de Tags

A nova tabela **tag_estacionamento** contém os seguintes atributos:

- **id_professor** (chave estrangeira para a tabela Docente);
- **tag** (código único da tag eletrônica);
- **status** (ativa ou inativa);
- **data_ativacao** (data de ativação da tag).

Isso permite que o acesso ao estacionamento seja feito automaticamente, reduzindo filas e aumentando a eficiência do sistema.

3.3. Atualização na Tabela de Pagamentos

Para contemplar o acordo com a bandeira **Visa**, a tabela de **pagamento** foi atualizada com um novo atributo:

- **isencao_visa** (booleano indicando se o professor tem direito aos 6 meses gratuitos).

Dessa forma, docentes que utilizam **Visa** podem usufruir do benefício por um semestre sem custos.

---

4. Benefícios das Atualizações

As mudanças proporcionaram vantagens significativas para o IDP e para os docentes:

- **Redução de filas**: eliminação do processo manual de reserva.
- **Automatização do acesso**: entrada e saída rápidas com as tags.
- **Facilidade nos pagamentos**: incentivo à adesão ao pagamento com Visa.
- **Melhor controle e segurança**: rastreamento eletrônico das tags ativas.

---

5. Conclusão

A transição para o sistema de tags de estacionamento e a parceria com a Visa representam avanços significativos para a gestão do estacionamento do IDP. A modelagem do banco de dados foi ajustada para atender às novas demandas, garantindo maior eficiência e comodidade aos docentes.

Com a remoção da tabela de reservas e a inclusão das tags eletrônicas, espera-se que o fluxo de entrada e saída seja otimizado, minimizando transtornos e melhorando a experiência acadêmica no campus.

Código SQL Atualizado:

```
CREATE TABLE IF NOT EXISTS vaga(
```

```
 id_vaga INT AUTO_INCREMENT NOT NULL,
 status_vaga ENUM("livre","ocupada"),
 tipo_vaga ENUM("prioritaria","comum"),
 localizacao VARCHAR(10),
 PRIMARY KEY(id_vaga)
);

CREATE TABLE IF NOT EXISTS tarifa(
 id_tarifa INT AUTO_INCREMENT NOT NULL,
 tipo_tarifa INT,
 valor DECIMAL,
 PRIMARY KEY(id_tarifa)
);

CREATE TABLE IF NOT EXISTS metodo_pagamento(
 id_metodo_pagamento INT AUTO_INCREMENT NOT NULL,
 tipo_metodo INT,
 descricao VARCHAR(50),
 PRIMARY KEY(id_metodo_pagamento)
);

CREATE TABLE IF NOT EXISTS usuario(
 id_usuario INT AUTO_INCREMENT NOT NULL,
 nome VARCHAR(50),
 cpf VARCHAR(20),
 tipo_usuario INT,
 PRIMARY KEY(id_usuario)
);

CREATE TABLE IF NOT EXISTS cadastro_cartao(
 id_cartao INT AUTO_INCREMENT NOT NULL,
 id_usuario INT,
 numero_cartao VARCHAR(20),
```

```
 validade DATE,
 cvv VARCHAR(5),
 bandeira VARCHAR(10),
 is_isento BOOLEAN,
 id_metodo_pagamento INT,
 PRIMARY KEY(id_cartao),
 FOREIGN KEY (id_metodo_pagamento) REFERENCES
metodo_pagamento(id_metodo_pagamento),
 FOREIGN KEY(id_usuario) REFERENCES usuario(id_usuario)
);

CREATE TABLE IF NOT EXISTS tag_estacionamento(
 id_tag INT AUTO_INCREMENT NOT NULL,
 id_usuario INT,
 id_veiculo INT,
 codigo_tag VARCHAR(20),
 ativo BOOLEAN,
 PRIMARY KEY(id_tag),
 FOREIGN KEY(id_usuario) REFERENCES
usuario(id_usuario),
 FOREIGN KEY(id_veiculo) REFERENCES veiculo(id_veiculo)
);

CREATE TABLE IF NOT EXISTS reserva(
 id_reserva INT,
 id_usuario INT,
 id_vaga INT,
 data_reserva DATETIME,
 PRIMARY KEY(id_reserva),
 FOREIGN KEY(id_usuario) REFERENCES
usuario(id_usuario),
 FOREIGN KEY(id_vaga) REFERENCES vaga(id_vaga)
);
```

```sql
CREATE TABLE IF NOT EXISTS veiculo(
 id_veiculo INT AUTO_INCREMENT NOT NULL,
 placa VARCHAR(10),
 modelo VARCHAR(20),
 cor VARCHAR(10),
 nome_proprietario VARCHAR(50),
 PRIMARY KEY(id_veiculo)
);

CREATE TABLE IF NOT EXISTS registro_entrada_saida(
 id_registro INT AUTO_INCREMENT NOT NULL,
 id_veiculo INT,
 id_vaga INT,
 id_usuario INT,
 id_tarifa INT,
 data_hora_entrada DATETIME,
 data_hora_saida DATETIME,
 tempo_permanencia INT,
 valor_pago DECIMAL,
 id_metodo_pagamento INT,

 PRIMARY KEY(id_registro),
 FOREIGN KEY(id_veiculo) REFERENCES
veiculo(id_veiculo),
 FOREIGN KEY(id_vaga) REFERENCES vaga(id_vaga),
 FOREIGN KEY(id_usuario) REFERENCES
usuario(id_usuario),
 FOREIGN KEY(id_tarifa) REFERENCES tarifa(id_tarifa),
 FOREIGN KEY(id_metodo_pagamento) REFERENCES
metodo_pagamento(id_metodo_pagamento)
);
```

```sql
SELECT nome FROM usuario a
INNER JOIN tag_estacionamento b ON a.id_usuario =
b.id_usuario
WHERE b.ativo = TRUE;

SELECT nome FROM usuario a
WHERE a.id_usuario IN(SELECT id_usuario FROM reserva)
AND a.id_usuario NOT IN(SELECT id_usuario FROM
tag_estacionamento);

SELECT a.id_usuario from usuario a
JOIN tag_estacionamento b ON a.id_usuario =
b.id_usuario
JOIN cadastro_cartao c ON c.id_usuario = a.id_usuario
WHERE b.ativo = TRUE
AND c.bandeira = 'VISA';

SELECT a.id_usuario from usuario a
JOIN tag_estacionamento b ON a.id_usuario =
b.id_usuario
WHERE b.ativo = FALSE;

SELECT * FROM cadastro_cartao a
WHERE a.bandeira = 'VISA'

SELECT a.nome from usuario a
JOIN tag_estacionamento b ON a.id_usuario =
b.id_usuario
JOIN cadastro_cartao c ON c.id_usuario = a.id_usuario
WHERE b.ativo = TRUE
```