

# Tecnologias para Análise e Desenvolvimento de Sistemas Trabalho Prático

Júlia Claudino Miranda

<sup>1</sup>Pontifícia Universidade Católica de Minas Gerais (PUC MINAS)

## 1. Tema - Sistema de Cobrança Automática em Estacionamentos do Shopping

O sistemas de cobrança automática adotam a tecnologia de radiofrequência para a identificação automática de veículos (IAV) com aplicações que vão desde a cobrança de pedágios, passando por controle automático de acesso e pedágio urbano até a inspeção veicular ou mesmo gerenciamento de tráfego em metrópoles. No Brasil, esse sistema não usado apenas para identificar frotas, mas também para consumo de serviços, como em estacionamentos de shoppings e aeroportos.

O pagamento automático nada mais é do que a intalação de um dispositivo eletrônico (Tag) no para-brisa do veículo que estando devidamente habilitado junto as empresas gestoras, possibilitára a passagem do usuário pela praça de pedágio ou estacionamentos, sem a necessidade de parar e manusear dinheiro.

Apesar deste sistema mostrar certa eficiência, ele ainda pode conter alguns problemas que devem ser levados em considerção. Como falhas técnicas, cobranças incorretas, congestionamentos nas entradas ou saídas de estacionamentos, dentro outros.

Embora estes serviços de cobrança automática tenham sido projetados para melhorar a eficiência e a satisfação das pessoas, é de sua importância abordar essas questões, a fim de assegurar a eficácia e a segurança do sistema para todas as partes envolvidas. A solução para a grande maioria desses desafios envolve a implementação cuidadosa, a manutenção regular, além da preocupação em proteger os dados dos usuários.

## 2. Diagrama Entidade e Relacionamento

Um diagrama entidade relacionamento é um tipo de fluxograma que ilustra como “entidades”, p. ex., pessoas, objetos ou conceitos, se relacionam entre si dentro de um sistema. Diagramas ER são mais utilizados para projetar ou depurar bancos de dados relacionais nas áreas de engenharia de software, sistemas de informações empresariais, educação e pesquisa.

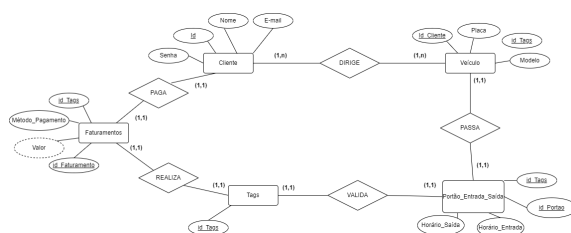


Figure 1. Diagrama

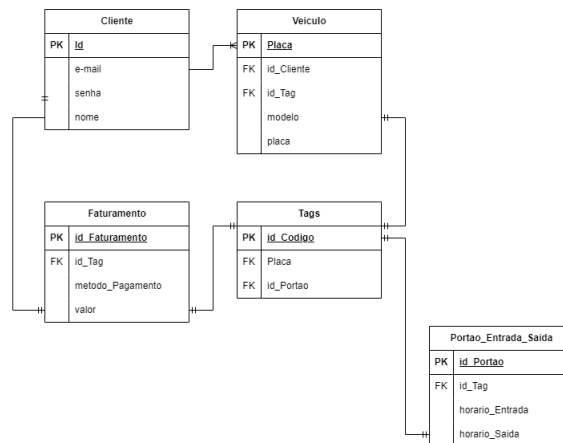


Figure 2. Fluxograma

### 3. Criação de Consultas - SQL

As imagens abaixo demonstram como foi realizada a consulta dos dados implementados pelo SQL.

#### 3.1. Junção de 3 Tabelas

```

/*
Abaixo 3 tabelas estão sendo combinadas utilizando o comando INNER JOIN
Ao final, será me entregue apenas:
NOME,CPF da tabela Clientes;
Modelo da tabela Veiculos;
Valor, MetodoPagamento da tabela Faturamentos
*/
SELECT Clientes.Nome, Clientes.CPF,
       Veiculos.Modelo,
       Faturamentos.Valor, Faturamentos.MetodoPagamento
FROM Clientes
INNER JOIN Veiculos
ON Clientes.Id = Veiculos.ClienteId
INNER JOIN Faturamentos
ON Clientes.Id = Faturamentos.ClienteId
  
```

Nome	CPF	Modelo	Valor	MetodoPagamento
Fernanda	12345678901	Fiat Siena	10	Débito
Júlia	98765432109	Honda City	20	Crédito
Leandro	23456789012	Ford KA	5	Débito
Ana	89012345678	Honda Civic	30	Crédito
Carlos	34567890123	Ford KA	25	Débito
Pedro	45678901234	BMW X1	15	Crédito

Figure 3. 3 Tabelas

A imagem demonstra como é realizada a Junção de 3 tabelas. Sendo elas **Clientes**, **Faturamentos** e **Veiculos**. Para fazer esta junção, foi utilizada a cláusula **INNER JOIN**, sendo ela muito útil para quando se é necessário combinar dados de diversas tabelas.

#### 3.2. Junção de 3 Tabelas - ORDER BY e WHERE

A imagem demonstra a junção de 3 tabelas e ao final é utilizado o **ORDER BY**, para classificar os resultados da consulta, enquanto o **WHERE** especifica as condições desejadas.

```

/*
Abaixo 3 tabelas estão sendo combinadas utilizando o comando INNER JOIN.
A cláusula WHERE me retorna apenas o nome,CPF,
modelo do veiculo e valor cobrado dos clientes cujo método
de pagamento escolhido foi 'Débito'.
Ao final,é solicitado que os resultados sejam organizados em
ordem alfabética a partir do ORDER BY
*/
SELECT Clientes.Nome,Clientes.CPF,
Veiculos.Modelo,
Faturamentos.Valor, Faturamentos.MetodoPagamento
FROM Clientes
INNER JOIN Veiculos
ON Clientes.Id = Veiculos.ClienteId
INNER JOIN Faturamentos
ON Clientes.Id = Faturamentos.ClienteId
WHERE Faturamentos.MetodoPagamento = 'Débito'
ORDER BY Nome ASC

```

	Nome	CPF	Modelo	Valor	MetodoPagamento
1	Carlos	34567890123	Ford KA	25	Debito
2	Fernanda	12345678901	Fiat Siena	10	Debito
3	Leandro	23456789012	Ford KA	5	Debito

Figure 4. ORDER BY e WHERE

### 3.3. Junção de 3 Tabelas - LIKE

```

/*
Abaixo 3 tabelas estão sendo combinadas utilizando o comando INNER JOIN.
A cláusula LIKE retorna como resultado apenas
os clientes que possuem um veiculo cujo modelo
começa com a letra 'H'
*/
SELECT Clientes.Nome,
Veiculos.Modelo,
Faturamentos.Valor, Faturamentos.MetodoPagamento
FROM Clientes
INNER JOIN Veiculos
ON Clientes.Id = Veiculos.ClienteId
INNER JOIN Faturamentos
ON Clientes.Id = Faturamentos.ClienteId
WHERE Veiculos.Modelo LIKE 'H%'

```

	Nome	Modelo	Valor	MetodoPagamento
1	Júlia	Honda City	20	Crédito
2	Ana	Honda Civic	30	Crédito

Figure 5. LIKE

A imagem demonstra a utilização da cláusula **LIKE**. **LIKE** é usada para realizar pesquisas "padronizadas", ou seja, caso seja necessário encontrar uma correspondência exata de um determinado termo, como palavras iniciadas ou terminadas com uma letra específica, o uso do **LIKE** é recomendado.

### 3.4. Junção de 3 Tabelas - IN

```
/*
Abaixo 3 tabelas estão sendo combinadas utilizando o comando INNER JOIN.
A cláusula IN retorna como resultado apenas
os clientes que pagaram valores exatos de 5, 15 e 20
*/
SELECT Clientes.Nome,
       Veiculos.Modelo,
       Faturamentos.Valor, Faturamentos.MetodoPagamento
FROM Clientes
INNER JOIN Veiculos
ON Clientes.Id = Veiculos.ClienteId
INNER JOIN Faturamentos
ON Clientes.Id = Faturamentos.ClienteId
WHERE Faturamentos.Valor IN (15, 20, 5)
```

Nome	Modelo	Valor	MetodoPagamento
Leandro	Ford KA	5	Débito
Pedro	BMW X1	15	Crédito
Júlia	Honda City	20	Crédito

Figure 6. IN

A imagem demonstra o método de utilização da cláusula **IN**, onde sua principal função é verificar se um valor específico corresponde a algum valor presente em uma lista ou subconsulta.

### 3.5. Junção de 2 Tabelas - GROUP BY e HAVING

```
/*
Abaixo 2 tabelas estão sendo combinadas utilizando o INNER JOIN.
A cláusula GROUP BY irá agrupar as linhas da tabela Valor e
me retornar apenas o valor máximo dos clientes cujo Id é
3, 6 e 7
*/
SELECT Nome, MAX(Valor) AS ValorMaximo
FROM Clientes
INNER JOIN Faturamentos
ON Clientes.Id = Faturamentos.ClienteId
WHERE Clientes.Id IN (3,6,7)
GROUP BY Clientes.Nome
```

Nome	ValorMaximo
Carlos	25
Júlia	20
Pedro	15

Figure 7. GROUP BY

A imagem demonstra o uso do **GROUP BY**. Esta cláusula é utilizada para agrupar os resultados de uma consulta com base nos valores de uma ou mais colunas.

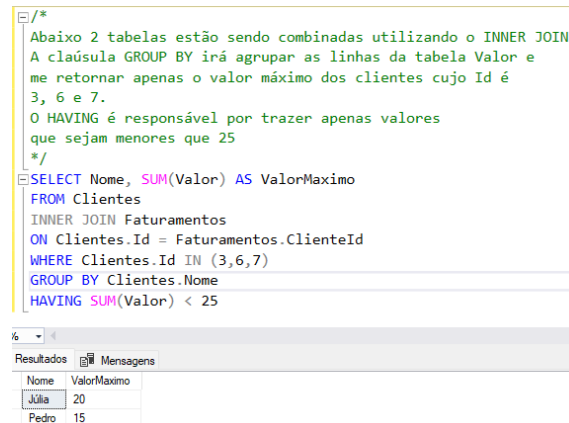


Figure 8. GROUP BY e HAVING

O **HAVING** se trata de uma cláusula que especifica quais registros agrupados são exibidos junto de uma cláusula **GROUP BY**.

### 3.6. Subselect SEM e COM Correlação

A imagem abaixo demonstra o uso da subconsulta correlacionada. Elas dependem e fazem referências as colunas de consultas externas a qual estão contidas.

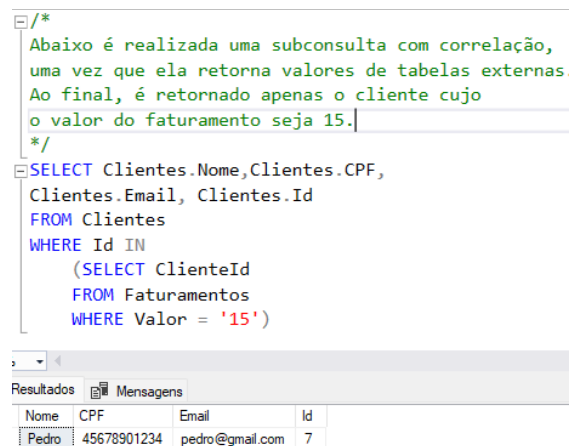


Figure 9. Correlação

Já a subconsulta **NÃO** correlacionada, se trata de uma consulta que não depende diretamente dos dados de uma consulta principal, ou seja, ela é independente.

```

/*
Abaixo é realizada uma subconsulta sem correlação
da tabela veiculos.
Ao final, é retornado apenas as informações do
veiculo cujo modelo começa com 'F'
*/
SELECT Veiculos.Modelo, Veiculos.Placa
FROM Veiculos
WHERE Modelo IN
(SELECT Modelo
FROM Veiculos
WHERE Modelo LIKE 'F%')

```

Modelo	Placa
Fiat Siena	ABC1234
Ford KA	DEF9012
Ford KA	GHI7890

Figure 10. Sem Correlação

### 3.7. Subselect com EXISTS

A imagem demonstra como é utilizada o **EXISTS**. Ela se trata de uma cláusula que testa quando há um ou mais resultados e uma subconsulta, retornando o valor **TRUE**. Dessa forma, é possível filtrar colunas dentro de uma subquery.

```

/*
Abaixo a cláusula EXISTS verifica se existe
ao menos uma linha na tabela Clientes que
corresponde ao número na tabela Veiculos.
Se a condição for verdadeira, o resultado apresentado será
o CPF e nome dos clientes
*/
SELECT Clientes.Nome, Clientes.CPF
FROM Clientes
WHERE EXISTS (SELECT Modelo
FROM Veiculos
WHERE Clientes.Id = Veiculos.ClienteId)

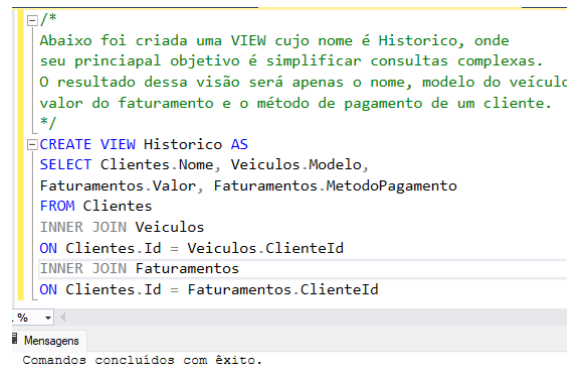
```

Nome	CPF
Fernanda	12345678901
Júlia	98765432109
Leandro	23456789012
Ana	89012345678
Carlos	34567890123
Pedro	45678901234

Figure 11. EXISTS

### 3.8. Criação de uma VIEW

A imagem mostra como é criada a **VIEW**. Esta cláusula pode ser definida como uma "tabela virtual", que é composta por linhas e colunas de dados vindos de tabelas relacionadas em uma query. Ao criar as views, é possível filtrar o conteúdo da tabela a ser exibida, tornando ela mais segura e simplificando os códigos.



```
/*
Abaixo foi criada uma VIEW cujo nome é Historico, onde
seu principal objetivo é simplificar consultas complexas.
O resultado dessa visão será apenas o nome, modelo do veículo,
valor do faturamento e o método de pagamento de um cliente.
*/
CREATE VIEW Historico AS
SELECT Clientes.Nome, Veiculos.Modelo,
Faturamentos.Valor, Faturamentos.MetodoPagamento
FROM Clientes
INNER JOIN Veiculos
ON Clientes.Id = Veiculos.ClienteId
INNER JOIN Faturamentos
ON Clientes.Id = Faturamentos.ClienteId
```

Mensagens  
Comandos concluídos com êxito.

Figure 12. VIEW HISTORICO

## 4. Consultas CRUD - MongoDB

O **CRUD** é um acrônimo que representa as quatro operações básicas em sistemas de gerenciamento de dados: Create, Read, Update e Delete. Ele descreve as principais funções para gerenciar informações em um banco de dados, permitindo a criação, leitura, atualização e exclusão de dados. Abaixo serão apresentadas as consultas utilizando estes métodos.

### 4.1. GET ALL

O **GET ALL** serve para recuperar e listar todos os registros ou documentos existentes em uma coleção de dados, permitindo a leitura de todos os itens.



Figure 13. GET ALL

## 4.2. GET ID

O **GET ID** serve para recuperar um registro específico com base no identificador único, ou seja o ID, permitindo a leitura dos detalhes desse registro .

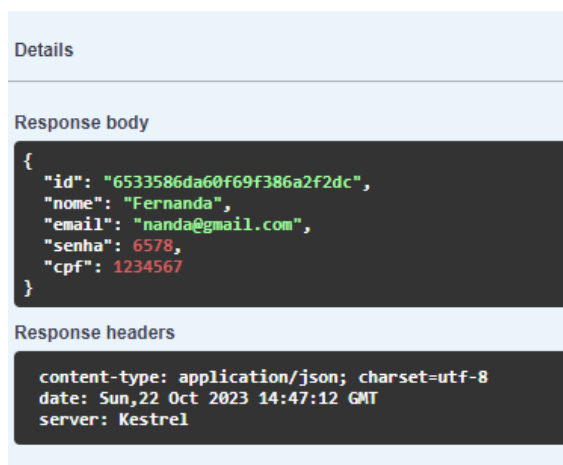


Figure 14. GET ID



### 4.3. POST

O **POST** serve para criar um novo registro em um banco de dados, ou seja, ele é usado para adicionar novos dados.

```
{
  "id": "6533586da60f69f386a2f2ff",
  "nome": "Daniel",
  "email": "daniel@gmail.com",
  "senha": 2453,
  "cpf": 2368749
}
```

Figure 15. Exemplo Post

```
Response body
{
  "id": "6533586da60f69f386a2f2ff",
  "nome": "Daniel",
  "email": "daniel@gmail.com",
  "senha": 2453,
  "cpf": 2368749
}

Response headers
content-type: application/json; charset=utf-8
date: Sun, 22 Oct 2023 14:59:06 GMT
location: https://localhost:7112/api/Sistema/6533586da60f69f386a2f2ff
server: Kestrel
```

Figure 16. POST

### 4.4. PUT

O **PUT** serve para atualizar ou modificar um registro existente em um banco de dados, com base em um ID.

```
Response body
{
  "id": "6533586da60f69f386a2f2de",
  "nome": "Leandro",
  "email": "leandro@gmail.com",
  "senha": 3527,
  "cpf": 2345678
}

Response headers
content-type: application/json; charset=utf-8
date: Sun, 22 Oct 2023 15:06:30 GMT
server: Kestrel
```

Figure 17. Exemplo Put

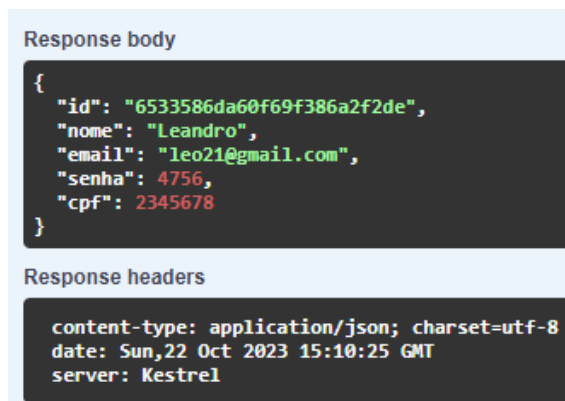


Figure 18. PUT

#### 4.5. DELETE

O **DELETE** serve para excluir permanentemente um registro ou entrada de um banco de dados.

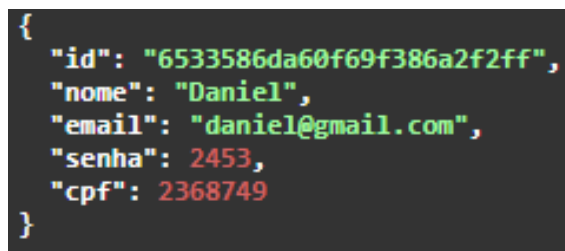


Figure 19. Exemplo Delete

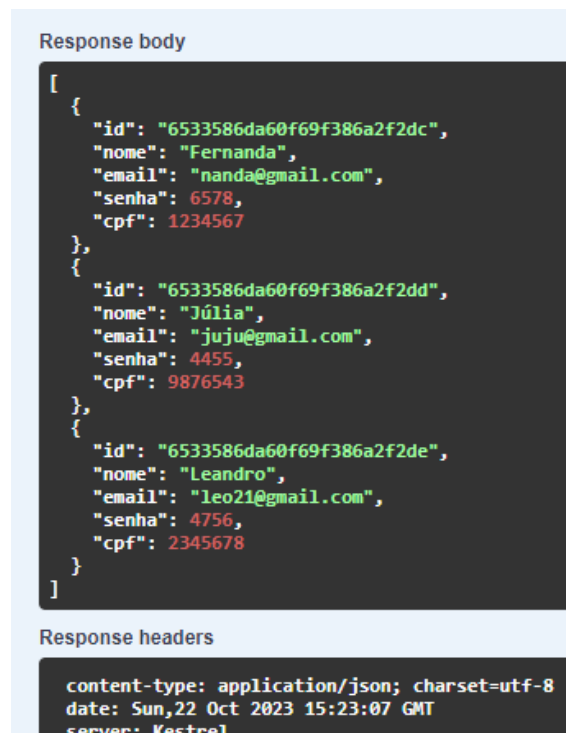


Figure 20. DELETE

## 5. Referências

Abaixo estão sendo identificados todas as referências bibliográficas utilizadas durante a planejamento do projeto.

### Links para mais informações:

>[https://www.em.com.br/app/noticia/tecnologia/2011/08/04/interna\\_tecnologia,243328/sistema-permite-cobranca-automatica-de-pedagio.shtml](https://www.em.com.br/app/noticia/tecnologia/2011/08/04/interna_tecnologia,243328/sistema-permite-cobranca-automatica-de-pedagio.shtml)

><https://www.rs.gov.br/carta-de-servicos/servicos?servico=1273#:~:text=O%20pagamento%20autom%C3%A1tico%20nada%20mais,%2C%20ainda%2C%20de%20manusear%20dinheiro.>

><https://www.lucidchart.com/pages/pt/o-que-ediagrama-entidade-relacionamento>