

---

# Documentação Técnica do Projeto de Banco de Dados

## Introdução ao Projeto

### Descrição do Projeto

O ManiClick é um site desenvolvido para facilitar a vida de uma manicure autônoma, oferecendo uma plataforma simples e intuitiva para divulgar seus serviços, atrair novos clientes e organizar seus agendamentos online. O projeto foi pensado para atender uma necessidade real de profissionais de bairro que ainda não possuem presença digital, unindo acessibilidade, usabilidade e funcionalidade. Com ele, os clientes podem visualizar serviços, preços, horários disponíveis e até agendar diretamente pelo site ou entrar em contato via WhatsApp, tudo de forma rápida e prática.

### Objetivo do Banco de Dados

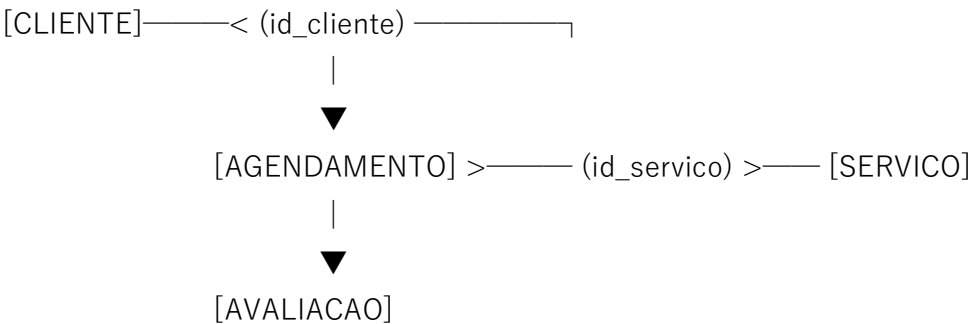
O objetivo do banco de dados para este projeto é centralizar e organizar todas as informações essenciais do sistema ManiClick, permitindo o armazenamento seguro e estruturado dos dados de clientes, serviços oferecidos, agendamentos realizados e avaliações feitas pelos clientes. Com isso, o banco de dados possibilita o gerenciamento eficiente dos atendimentos, o controle dos horários e status dos agendamentos, a análise da satisfação dos clientes por meio das avaliações e o acompanhamento da popularidade dos serviços. Dessa forma, o banco de dados serve como a base para o funcionamento do sistema, garantindo integridade, facilidade de consulta e atualização das informações, além de apoiar a tomada de decisões e o crescimento do negócio de forma profissional e confiável.

---

## Modelagem de Dados

### Diagrama Entidade-Relacionamento (DER)

O Diagrama Entidade-Relacionamento (DER) ilustra as entidades e relacionamentos do sistema, como mostrado abaixo:



---

## Estrutura do Banco de Dados

### Scripts de Criação do Banco de Dados

```
-- Listar todos os agendamentos futuros
SELECT a.id_agendamento, c.nome AS cliente, s.nome AS servico, a.data, a.hora, a.status
FROM agendamento a
JOIN cliente c ON a.id_cliente = c.id_cliente
JOIN servico s ON a.id_servico = s.id_servico
WHERE a.data >= CURDATE();

-- Listar clientes e suas avaliações
SELECT c.nome, a.nota, a.comentario
FROM avaliacao a
JOIN cliente c ON a.id_cliente = c.id_cliente;

-- Serviços mais agendados
SELECT s.nome, COUNT(*) AS total_agendamentos
FROM agendamento a
JOIN servico s ON a.id_servico = s.id_servico
GROUP BY s.nome
ORDER BY total_agendamentos DESC;

-- Agendamentos por status
SELECT status, COUNT(*) AS quantidade
FROM agendamento
GROUP BY status;

-- Criação das tabelas principais

CREATE TABLE cliente (
    id_cliente INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    telefone VARCHAR(20),
    email VARCHAR(100)
);

CREATE TABLE servico (
    id_servico INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    preco DECIMAL(8,2) NOT NULL,
    duracao_minutos INT NOT NULL
);

CREATE TABLE agendamento (
    id_agendamento INT PRIMARY KEY AUTO_INCREMENT,
    id_cliente INT,
    id_servico INT,
    data DATE NOT NULL,
    hora TIME NOT NULL,
    status VARCHAR(20) DEFAULT 'Pendente',
    observacao VARCHAR(255),
    FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente),
    FOREIGN KEY (id_servico) REFERENCES servico(id_servico)
);

CREATE TABLE avaliacao (
    id_avaliacao INT PRIMARY KEY AUTO_INCREMENT,
    id_cliente INT,
    nota INT CHECK (nota BETWEEN 1 AND 5),
    comentario VARCHAR(255),
    data_avaliacao DATE,
    FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente)
);
```

```
-- Clientes
INSERT INTO cliente (nome, telefone, email) VALUES
('Ana Paula', '19999999999', 'ana@email.com'),
('Bruna Silva', '19988888888', 'bruna@email.com'),
('Carla Mendes', '19977777777', 'carla@email.com');

-- Serviços
INSERT INTO servico (nome, preco, duracao_minutos) VALUES
('Esmaltação simples', 20.00, 30),
('Francesinha', 25.00, 40),
('Unha decorada', 30.00, 50),
('Pé e mão', 45.00, 60);

-- Agendamentos
INSERT INTO agendamento (id_cliente, id_servico, data, hora, status, observacao) VALUES
(1, 1, '2024-06-10', '10:00', 'Confirmado', ''),
(2, 2, '2024-06-11', '14:00', 'Pendente', 'cliente pediu francesinha especial'),
(3, 3, '2024-06-12', '16:00', 'Cancelado', 'Cancelado pelo cliente');

-- Avaliações
INSERT INTO avaliacao (id_cliente, nota, comentario, data_avaliacao) VALUES
(1, 5, 'ótimo atendimento!', '2024-06-10'),
(2, 4, 'Gostei muito!', '2024-06-11');
```

Descrição das Tabelas

• Tabela: Cliente

Campo	Tipo	Descrição
id_cliente	INT, PK, AI	Identificador do cliente
nome	VARCHAR(100)	Nome do cliente
telefone	VARCHAR(20)	Telefone do cliente
email	VARCHAR(100)	Email do cliente

• Tabela: Serviço

Campo	Tipo	Descrição
id_servico	INT, PK, AI	Identificador do serviço
nome	VARCHAR(100)	Nome do serviço
preco	DECIMAL(8,2)	Preço do serviço
duracao_minutos	INT	Duração em minutos

• Tabela: Agendamento

Campo	Tipo	Descrição
id_agendamento	INT, PK, AI	Identificador do agendamento
id_cliente	INT, FK	Cliente que fez o agendamento
id_servico	INT, FK	Serviço agendado
data	DATE	Data do agendamento
hora	TIME	Hora do agendamento
status	VARCHAR(20)	Status (Pendente, Confirmado...)
observacao	VARCHAR(255)	Observações

• Tabela: Avaliação

Campo	Tipo	Descrição
id_avaliacao	INT, PK, AI	Identificador da avaliação
id_cliente	INT, FK	Cliente que avaliou
nota	INT	Nota (1 a 5)
comentario	VARCHAR(255)	Comentário
data_avaliacao	DATE	Data da avaliação

---

## Consultas e Funcionalidades

### Descrição das Consultas

- 1- **vw\_agendamentos\_detalhados**: Exibe nome do cliente, serviço, data, hora e status dos agendamentos.

```
CREATE VIEW vw_agendamentos_detalhados AS
SELECT
    a.id_agendamento,
    c.nome AS cliente,
    s.nome AS servico,
    a.data,
    a.hora,
    a.status
FROM agendamento a
JOIN cliente c ON a.id_cliente = c.id_cliente
JOIN servico s ON a.id_servico = s.id_servico;
```

- 2- **vw\_clientes\_avaliacoes**: Lista as avaliações feitas por cada cliente.

```
CREATE VIEW vw_clientes_avaliacoes AS
SELECT
    c.nome AS cliente,
    a.nota,
    a.comentario,
    a.data_avaliacao
FROM avaliacao a
JOIN cliente c ON a.id_cliente = c.id_cliente;
```

- 3- **vw\_servicos\_populares**: Mostra a quantidade de agendamentos por serviço.

```
CREATE VIEW vw_servicos_populares AS
SELECT
    s.nome AS servico,
    COUNT(a.id_agendamento) AS total_agendamentos
FROM servico s
LEFT JOIN agendamento a ON s.id_servico = a.id_servico
GROUP BY s.nome;
```

- 4- **vw\_proximos\_confirmados**: Lista os agendamentos confirmados a partir da data atual.

```
CREATE VIEW vw_proximos_confirmados AS
SELECT
    a.id_agendamento,
    c.nome AS cliente,
    s.nome AS servico,
    a.data,
    a.hora,
    a.status
FROM agendamento a
JOIN cliente c ON a.id_cliente = c.id_cliente
JOIN servico s ON a.id_servico = s.id_servico
WHERE a.status = 'Confirmado' AND a.data >= CURDATE();
```

---

---

## Descrição das Funcionalidades

1- **sp\_inserir\_cliente**: Insere um novo cliente no sistema.

```
DELIMITER $$
CREATE PROCEDURE sp_inserir_cliente(
    IN p_nome VARCHAR(100),
    IN p_telefone VARCHAR(20),
    IN p_email VARCHAR(100)
)
BEGIN
    INSERT INTO cliente (nome, telefone, email)
    VALUES (p_nome, p_telefone, p_email);
END $$
DELIMITER ;
```

2- **sp\_cancelar\_agendamento**: Atualiza o status de um agendamento para "Cancelado".

```
DELIMITER $$
CREATE PROCEDURE sp_cancelar_agendamento(
    IN p_id_agendamento INT
)
BEGIN
    UPDATE agendamento
    SET status = 'Cancelado'
    WHERE id_agendamento = p_id_agendamento;
END $$
DELIMITER ;
```

3- **fn\_total\_agendamentos\_cliente**: Retorna o número total de agendamentos de um cliente.

```
DELIMITER $$
CREATE FUNCTION fn_total_agendamentos_cliente(p_id_cliente INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total INT;
    SELECT COUNT(*) INTO total
    FROM agendamento
    WHERE id_cliente = p_id_cliente;
    RETURN total;
END $$
DELIMITER ;
```

4- **fn\_media\_avaliacoes**: Retorna a média das notas das avaliações.

```
DELIMITER $$
CREATE FUNCTION fn_media_avaliacoes()
RETURNS DECIMAL(3,2)
DETERMINISTIC
BEGIN
    DECLARE media DECIMAL(3,2);
    SELECT AVG(nota) INTO media FROM avaliacao;
    RETURN media;
END $$
DELIMITER ;
```

5- **trg\_valida\_nota\_avaliacao**: Garante que a nota atribuída esteja entre 1 e 5.

```
DELIMITER $$
CREATE TRIGGER trg_valida_nota_avaliacao
BEFORE INSERT ON avaliacao
FOR EACH ROW
BEGIN
    IF NEW.nota < 1 OR NEW.nota > 5 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Nota deve ser entre 1 e 5';
    END IF;
END $$
DELIMITER ;
```

6- **trg\_cancelar\_agendamento**: Ao cancelar um agendamento, registra uma observação com data/hora.

```
DELIMITER $$
CREATE TRIGGER trg_cancelar_agendamento
BEFORE UPDATE ON agendamento
FOR EACH ROW
BEGIN
    IF NEW.status = 'Cancelado' AND OLD.status <> 'Cancelado' THEN
        SET NEW.observacao = CONCAT(
            IFNULL(OLD.observacao, ''),
            ' [Cancelado em ', DATE_FORMAT(NOW(), '%Y-%m-%d %H:%i:%s'), ']'
        );
    END IF;
END $$
DELIMITER ;
```

## Exemplos de Uso

### Inserir um novo cliente:

```
CALL sp_inserir_cliente('Maria Teste', '19912345678', 'maria@email.com');
```

### Agendar um serviço:

```
INSERT INTO agendamento (id_cliente, id_servico, data, hora, status)
VALUES (1, 2, '2024-06-20', '15:00', 'Pendente');
```

### Cancelar um agendamento:

```
CALL sp_cancelar_agendamento(3);
```

### Ver agendamentos detalhados:

```
SELECT * FROM vw_agendamentos_detalhados;
```

### Ver serviços mais populares:

```
SELECT * FROM vw_servicos_populares;
```

### Ver média das avaliações:

```
SELECT fn_media_avaliacoes();
```