Documentação Técnica do Projeto de Banco de Dados

Introdução ao Projeto

Descrição do Projeto

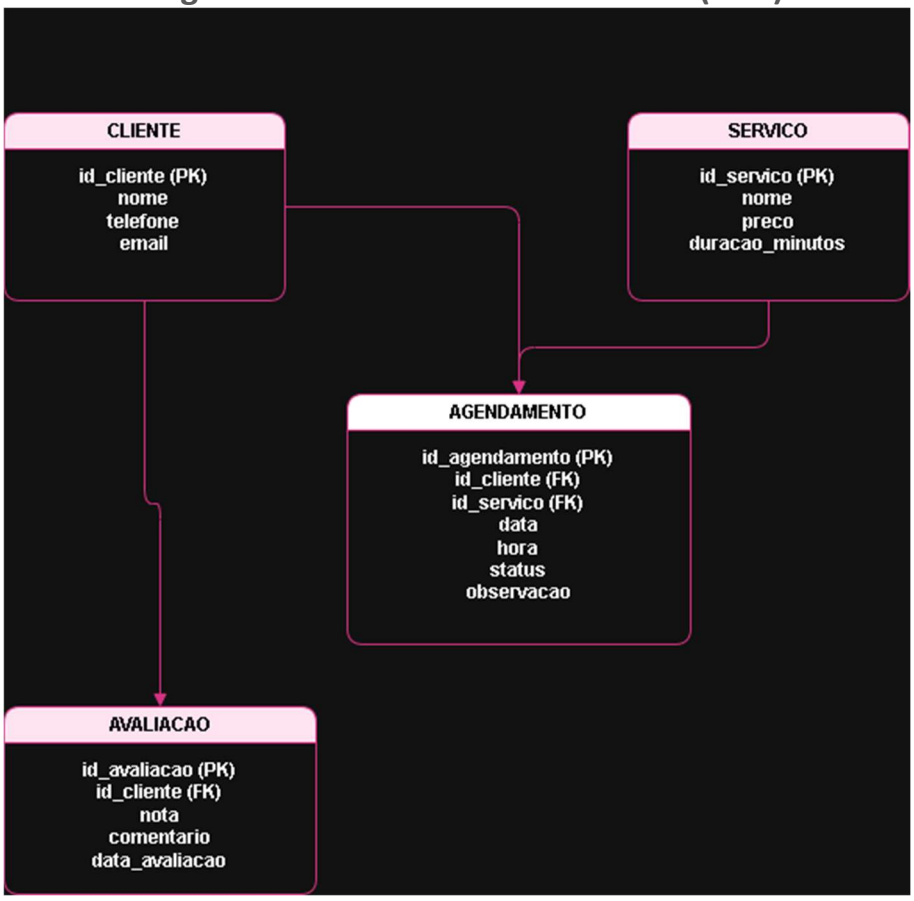
O **ManiClick** é uma plataforma web desenvolvida para facilitar a rotina de manicures autônomas, oferecendo um ambiente digital simples e intuitivo para divulgação de serviços, atração de novos clientes e organização de agendamentos online. O sistema foi projetado para profissionais que buscam presença digital, unindo acessibilidade, usabilidade e funcionalidades práticas. Os clientes podem visualizar serviços, preços, horários disponíveis, agendar diretamente pelo site e entrar em contato via WhatsApp, tudo de forma rápida e eficiente.

Objetivo do Banco de Dados

O objetivo do banco de dados do ManiClick é centralizar e organizar todas as informações essenciais do sistema, permitindo o armazenamento seguro e estruturado dos dados de clientes, serviços, agendamentos e avaliações. O banco de dados possibilita o gerenciamento eficiente dos atendimentos, controle de horários e status dos agendamentos, análise da satisfação dos clientes por meio das avaliações e acompanhamento da popularidade dos serviços. Assim, garante integridade, facilidade de consulta e atualização das informações, apoiando a tomada de decisões e o crescimento do negócio.

Modelagem de Dados

Diagrama Entidade-Relacionamento (DER)



Estrutura do Banco de Dados

Scripts de Criação do Banco de Dados

```sql
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)
);
```

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

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;
```

vw_clientes_avaliacoes: Lista as avaliações feitas por cada cliente.

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

vw_servicos_populares: Mostra a quantidade de agendamentos por serviço.

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

vw_proximos_confirmados: Lista os agendamentos confirmados a partir da data atual.

```
CREATE VIEW vw_proximos_confirmados AS
SELECT a.*, c.nome AS cliente, s.nome AS servico
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: Procedures, Funções e Triggers

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 ;
```

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 ;
```

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 ;
```

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 ;
```

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 ;
```

trg_cancelar_agendamento: Ao cancelar um agendamento, registra uma observação com data/hora.

```
DELIMITER $$
CREATE TRIGGER trg_cancelar_agendamento
BEFORE UPDATE ON agendamento
FOR EACH ROW
BEGIN
    IF NEW.status = 'Cancelado' AND OLD.status <> 'Cancelado' THEN
        SET NEW.observacao = CONCAT(OLD.observacao, ' [Cancelado em ', NOW(), ']');
    END IF;
END $$
DELIMITER ;
```

Exemplos de Uso

Inserir um novo cliente:

```
CALL sp_inserir_cliente('Maria Souza', '19999999999', 'maria@email.com');
```

Agendar um serviço:

```
INSERT INTO agendamento (id_cliente, id_servico, data, hora, status)
VALUES (1, 2, '2024-07-01', '14:00', 'Confirmado');
```

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();
```

Considerações Finais

O banco de dados do ManiClick foi projetado para garantir integridade, segurança e facilidade de manutenção, apoiando todas as funcionalidades do sistema web. A modelagem relacional permite consultas eficientes, relatórios gerenciais e integração com o frontend, promovendo o crescimento do negócio e a satisfação dos usuários.