Projeto de Banco de Dados - Projeto Integrador (PI)

1. Introdução

Este documento descreve o projeto de banco de dados para o **Projeto Integrador (PI)**, desenvolvido com o objetivo de gerenciar **agendamentos de serviços realizados por profissionais para clientes**. O sistema permitirá o gerenciamento de dados de clientes, profissionais, serviços, agendamentos, feedbacks e relatórios.

O banco de dados foi modelado de acordo com as necessidades de um sistema de agendamento de serviços, envolvendo a criação de tabelas, visões (views), consultas (SELECT), procedimentos armazenados (procedures), funções e triggers, a fim de garantir a integridade, segurança e desempenho das operações.

2. Minimundo do Projeto

O **minimundo** do projeto refere-se à descrição dos objetos de interesse e suas interações no domínio do sistema. Para este projeto, o minimundo envolve os seguintes componentes:

- **Clientes**: Pessoas que agendam serviços com os profissionais cadastrados no sistema.
- **Profissionais**: Pessoas que prestam serviços, como cabeleireiros, manicures, barbeiros, etc.
- **Serviços**: Tipos de serviços oferecidos pelos profissionais, como corte de cabelo, manicure, etc.
- **Agendamentos**: Registros de horários agendados pelos clientes para realizar um ou mais serviços.
- **Feedbacks**: Avaliações fornecidas pelos clientes após a realização de um serviço.
- **Horários**: Definem a disponibilidade de cada profissional para atender aos clientes.

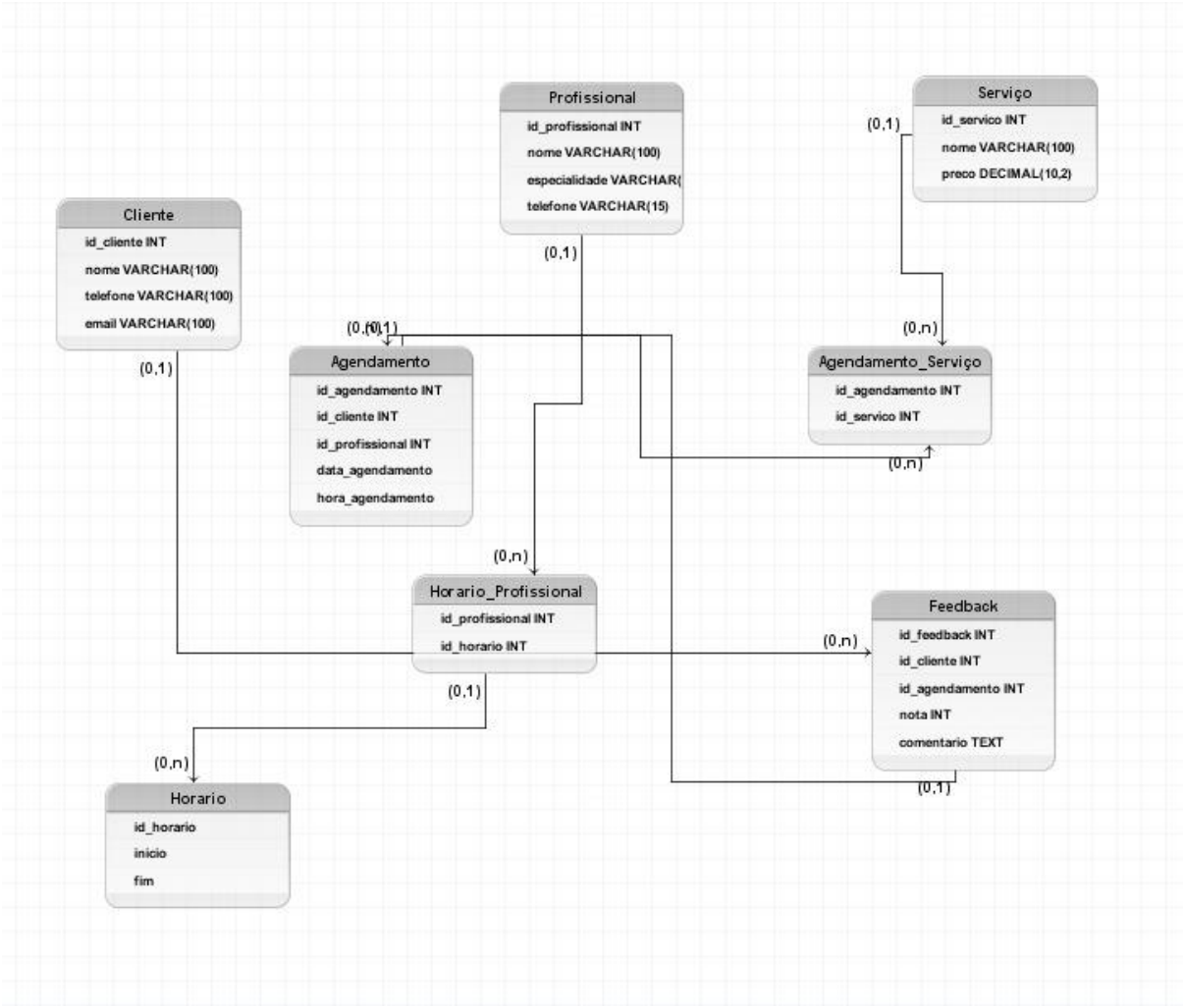
3. Modelo Conceitual (Diagrama MER)

O Modelo Entidade-Relacionamento (MER) descreve as entidades, seus atributos e os relacionamentos entre elas. Inclui as entidades **Cliente**, **Profissional**, **Serviço**, **Agendamento**, **Feedback**, e **Horário**, além de suas associações.

Relacionamentos principais:

- **Cliente → Agendamento → Profissional**: Um cliente pode fazer vários agendamentos, e cada agendamento envolve um profissional específico.
- **Agendamento → Serviço**: Cada agendamento pode envolver um ou mais serviços prestados.
- **Cliente → Feedback → Agendamento**: Após o serviço, o cliente pode fornecer feedback sobre o agendamento.

Diagrama MER



.

4. Modelo Lógico (Diagrama MR)

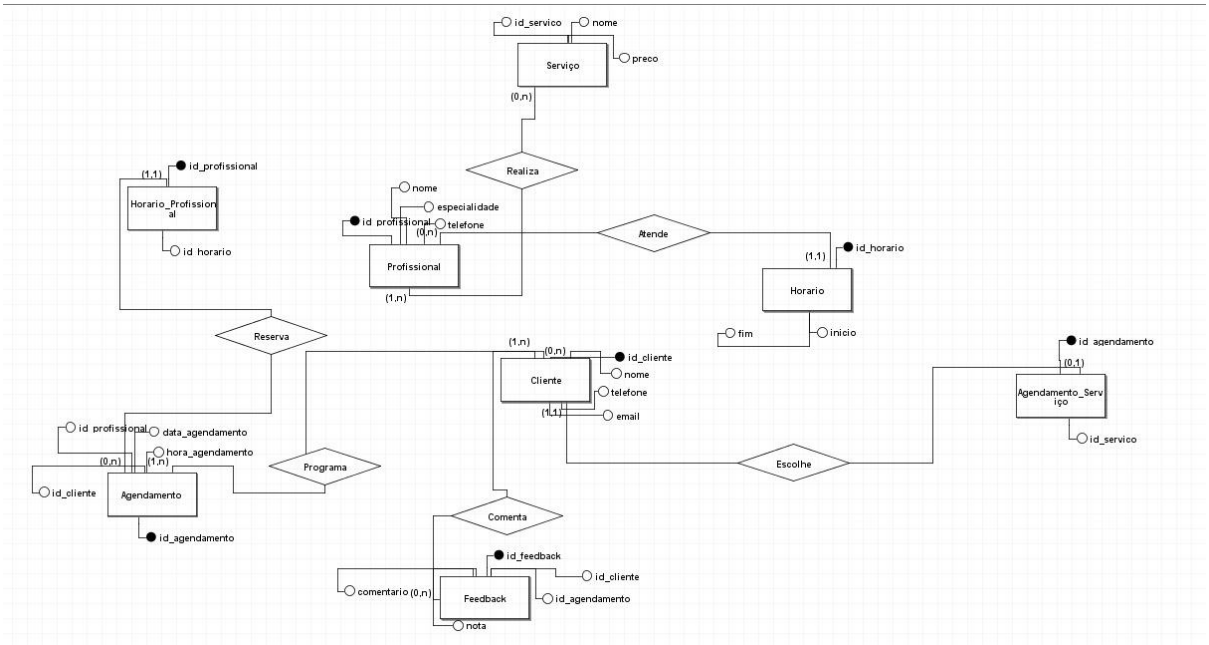
O Modelo Relacional (MR) descreve como o modelo conceitual é mapeado para um banco de dados relacional. As entidades do MER são convertidas em tabelas com as relações representadas por chaves primárias e estrangeiras.

Tabelas principais:

1. **Cliente**: Tabela que armazena informações dos clientes.
2. **Profissional**: Tabela que armazena dados dos profissionais.
3. **Serviço**: Tabela que descreve os serviços disponíveis.
4. **Agendamento**: Tabela que registra os agendamentos realizados pelos clientes.
5. **Agendamento_Serviço**: Tabela associativa entre agendamentos e serviços.
6. **Feedback**: Tabela para registrar feedbacks de clientes sobre os serviços.
7. **Horário**: Tabela que define os horários de disponibilidade dos profissionais.

8. **Horario_Profissional**: Tabela associativa entre profissionais e seus horários.

Diagrama MR



5. Scripts de Criação (DDL)

-- Tabela Cliente CREATE TABLE Cliente (id_cliente INT AUTO_INCREMENT PRIMARY KEY, nome VARCHAR(100) NOT NULL, telefone VARCHAR(15), email VARCHAR(100) UNIQUE, data_cadastro DATE DEFAULT CURRENT_DATE);

-- Tabela Profissional CREATE TABLE Profissional (id_profissional INT AUTO_INCREMENT PRIMARY KEY, nome VARCHAR(100) NOT NULL, especialidade VARCHAR(100), telefone VARCHAR(15), email VARCHAR(100) UNIQUE, data_cadastro DATE DEFAULT CURRENT_DATE);

-- Tabela Horário CREATE TABLE Horario (id_horario INT AUTO_INCREMENT PRIMARY KEY, inicio TIME NOT NULL, fim TIME NOT NULL);

-- Tabela Horário_Profissional (associativa entre Profissional e Horário) CREATE TABLE Horario_Profissional (id_profissional INT, id_horario INT, PRIMARY KEY (id_profissional, id_horario), FOREIGN KEY (id_profissional) REFERENCES Profissional(id_profissional) ON DELETE CASCADE, FOREIGN KEY (id_horario) REFERENCES Horario(id_horario) ON DELETE CASCADE);

-- Tabela Serviço CREATE TABLE Servico (id_servico INT AUTO_INCREMENT PRIMARY KEY, nome VARCHAR(100) NOT NULL, preco DECIMAL(10, 2) NOT NULL);

-- Tabela Agendamento CREATE TABLE Agendamento (id_agendamento INT AUTO_INCREMENT PRIMARY KEY, id_cliente INT, id_profissional INT, data_agendamento DATE NOT NULL, hora_agendamento TIME NOT NULL, status ENUM('PENDENTE', 'CONCLUÍDO', 'CANCELADO') DEFAULT 'PENDENTE', valor_total DECIMAL(10, 2), -- Adicionada coluna valor_total FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente) ON DELETE CASCADE, FOREIGN KEY (id_profissional) REFERENCES Profissional(id_profissional) ON DELETE CASCADE);

-- Tabela Agendamento_Servico (associativa entre Agendamento e Serviço) CREATE TABLE Agendamento_Servico (id_agendamento INT, id_servico INT, PRIMARY KEY (id_agendamento, id_servico), FOREIGN KEY (id_agendamento) REFERENCES Agendamento(id_agendamento) ON DELETE CASCADE, FOREIGN KEY (id_servico) REFERENCES Servico(id_servico) ON DELETE CASCADE);

-- Tabela Feedback CREATE TABLE Feedback (id_feedback INT AUTO_INCREMENT PRIMARY KEY, id_cliente INT, id_agendamento INT, comentario TEXT, nota INT CHECK (nota BETWEEN 1 AND 5), data_feedback DATE DEFAULT CURRENT_DATE, FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente) ON DELETE CASCADE, FOREIGN KEY (id_agendamento) REFERENCES Agendamento(id_agendamento) ON DELETE CASCADE);

-- Tabela Notificacoes CREATE TABLE Notificacoes (id_notificacao INT AUTO_INCREMENT PRIMARY KEY, id_cliente INT, mensagem TEXT, data_notificacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP, FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente) ON DELETE CASCADE);

-- Tabela Log_Status_Agendamento CREATE TABLE Log_Status_Agendamento (id_log INT AUTO_INCREMENT PRIMARY KEY, id_agendamento INT, status_antigo ENUM('PENDENTE', 'CONCLUÍDO', 'CANCELADO'), status_novo ENUM('PENDENTE', 'CONCLUÍDO', 'CANCELADO'), data_alteracao TIMESTAMP DEFAULT CURRENT_TIMESTAMP, FOREIGN KEY (id_agendamento) REFERENCES Agendamento(id_agendamento) ON DELETE CASCADE);

-- Inserindo 10 clientes INSERT INTO Cliente (nome, telefone, email) VALUES ('Ana Maria', '11999998888', 'ana.maria@email.com'), ('Carlos Silva', '11988887777', 'carlos.silva@email.com'), ('Mariana Santos', '11977776666', 'mariana.santos@email.com'), ('Joana Almeida', '11966664444', 'joana.almeida@email.com'), ('Pedro Costa', '11955553333', 'pedro.costa@email.com'), ('Fernanda Lima', '11944442222',

'fernanda.lima@email.com'), ('Luiz Carvalho', '11933331111', 'luiz.carvalho@email.com'), ('Patricia Andrade', '11922220000', 'patricia.andrade@email.com'), ('Rafael Souza', '11911119999', 'rafael.souza@email.com'), ('Julia Gomes', '11900008888', 'julia.gomes@email.com');

-- Inserindo 10 profissionais INSERT INTO Profissional (nome, especialidade, telefone, email) VALUES ('João Pereira', 'Cabeleireiro', '11966665555', 'joao.pereira@email.com'), ('Paula Oliveira', 'Manicure', '11955554444', 'paula.oliveira@email.com'), ('Roberta Souza', 'Esteticista', '11944443333', 'roberta.souza@email.com'), ('Cláudia Mendes', 'Cabeleireira', '11933332222', 'claudia.mendes@email.com'), ('Fernando Torres', 'Barbeiro', '11922221111', 'fernando.torres@email.com'), ('Tatiana Braga', 'Maquiadora', '11911110000', 'tatiana.braga@email.com'), ('Ronaldo Alves', 'Massoterapeuta', '11900007777', 'ronaldo.alves@email.com'), ('Gabriela Araújo', 'Colorista', '11899996666', 'gabriela.araujo@email.com'), ('Lucas Vieira', 'Depilador', '11888885555', 'lucas.vieira@email.com'), ('Sandra Ferreira', 'Pedicure', '11877774444', 'sandra.ferreira@email.com');

-- Criando Views

-- View agendamentos completos CREATE VIEW vw_agendamentos_completos AS SELECT Agendamento.id_agendamento, Cliente.nome AS cliente, Profissional.nome AS profissional, Agendamento.data_agendamento, Agendamento.hora_agendamento, Agendamento.status FROM Agendamento JOIN Cliente ON Agendamento.id_cliente = Cliente.id_cliente JOIN Profissional ON Agendamento.id_profissional = Profissional.id_profissional;

-- View serviços por agendamento CREATE VIEW vw_servicos_por_agendamento AS SELECT Agendamento.id_agendamento, Servico.nome AS servico, Servico.preco FROM Agendamento_Servico JOIN Servico ON Agendamento_Servico.id_servico = Servico.id_servico;

-- View agendamentos de cliente CREATE VIEW vw_agendamentos_cliente AS SELECT A.id_agendamento, C.nome AS cliente, P.nome AS profissional, A.data_agendamento, A.hora_agendamento, A.status FROM Agendamento A JOIN Cliente C ON A.id_cliente = C.id_cliente JOIN Profissional P ON A.id_profissional = P.id_profissional WHERE C.id_cliente = :id_cliente;

-- View feedbacks de cliente CREATE VIEW vw_feedbacks_cliente AS SELECT C.nome AS cliente, A.id_agendamento, F.nota, F.comentario, F.data_feedback FROM Feedback F JOIN Agendamento A ON F.id_agendamento = A.id_agendamento JOIN Cliente C ON F.id_cliente = C.id_cliente;

-- View de profissionais CREATE VIEW vw_profissionais AS SELECT P.id_profissional, P.nome, P.especialidade FROM Profissional P;

-- View de serviços CREATE VIEW vw_servicos AS SELECT S.id_servico, S.nome, S.preco FROM Servico S;

-- Criando Triggers

-- Trigger para atualização de data quando agendamento for concluído CREATE TRIGGER trg_agendamento_concluido AFTER UPDATE ON Agendamento FOR EACH ROW BEGIN IF NEW.status = 'CONCLUÍDO' THEN UPDATE Agendamento SET data_agendamento = CURRENT_DATE WHERE id_agendamento = NEW.id_agendamento; END IF; END;

-- Trigger para verificar conflito de horário de agendamento CREATE TRIGGER trg_check_conflito_agendamento BEFORE INSERT ON Agendamento FOR EACH ROW BEGIN DECLARE agendamento_existente INT; SELECT COUNT(*) INTO agendamento_existente FROM Agendamento WHERE id_profissional = NEW.id_profissional AND data_agendamento = NEW.data_agendamento AND hora_agendamento = NEW.hora_agendamento AND status = 'PENDENTE';

```
IF agendamento_existente > 0 THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Horário de
agendamento já reservado!';
END IF;
```

END;

-- Trigger para log de alteração de status de agendamento CREATE TRIGGER trg_log_alteracao_status_agendamento AFTER UPDATE ON Agendamento FOR EACH ROW BEGIN IF OLD.status != NEW.status THEN INSERT INTO Log_Status_Agendamento (id_agendamento, status_antigo, status_novo, data_alteracao) VALUES (NEW.id_agendamento, OLD.status, NEW.status, CURRENT_TIMESTAMP); END IF; END;

-- Trigger para cálculo do valor total do agendamento CREATE TRIGGER trg_calcular_valor_agendamento AFTER INSERT ON Agendamento_Servico FOR EACH ROW BEGIN DECLARE total DECIMAL(10,2); SELECT SUM(Servico.preco) INTO total FROM Servico JOIN Agendamento_Servico ON Servico.id_servico = Agendamento_Servico.id_servico WHERE Agendamento_Servico.id_agendamento = NEW.id_agendamento;

```
UPDATE Agendamento
SET valor_total = total
WHERE id_agendamento = NEW.id_agendamento;


END;
```

-- Trigger para validar a nota de feedback CREATE TRIGGER trg_validar_nota_feedback BEFORE INSERT ON Feedback FOR EACH ROW BEGIN IF NEW.nota < 1 OR NEW.nota > 5 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A nota deve estar entre 1 e 5!'; END IF; END;

-- Trigger para notificação de cancelamento de agendamento CREATE TRIGGER trg_notificar_cancelamento AFTER UPDATE ON Agendamento FOR EACH ROW BEGIN IF NEW.status = 'CANCELADO' THEN INSERT INTO Notificacoes (id_cliente, mensagem, data_notificacao) VALUES (NEW.id_cliente, 'Seu agendamento foi cancelado.', CURRENT_TIMESTAMP); END IF; END;