Documentação Técnica do Projeto de Banco de Dados

Introdução ao Projeto

Descrição do Projeto

O projeto consiste no desenvolvimento de uma aplicação web utilizando HTML, CSS e JavaScript, com o objetivo de criar um gerador de orçamentos (FORC – Ferramenta de Orçamento Rápido e Cálculo) que permita a empresas ou prestadores de serviço criarem orçamentos de forma rápida, padronizada e eficiente.

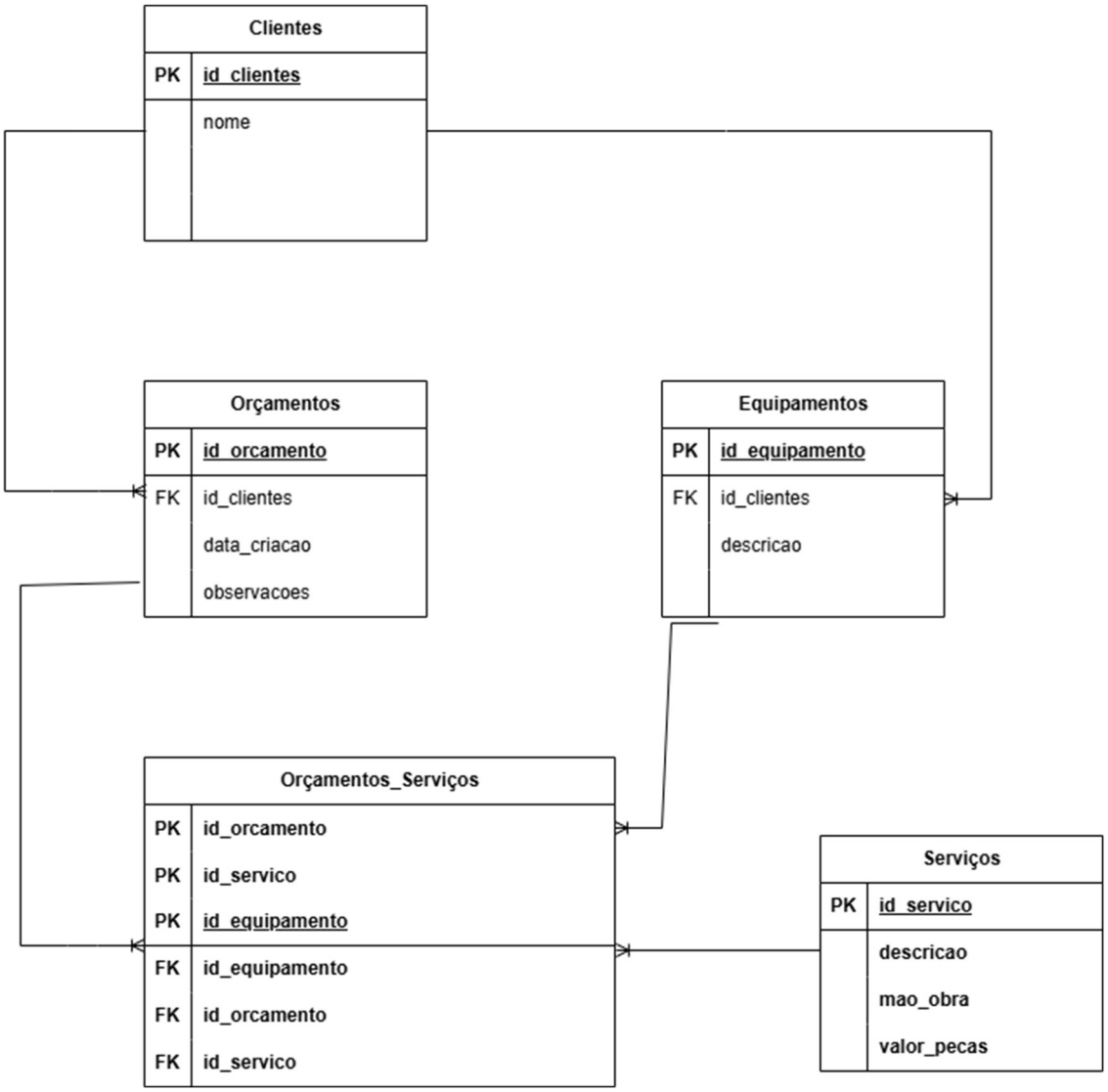
Objetivo do Banco de Dados

O objetivo do banco de dados é organizar e gerenciar informações sobre orçamentos, clientes, serviços e custos, garantindo eficiência no processamento de transações, cálculos e relatórios.

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

Os scripts SQL para criação das tabelas e seus relacionamentos são apresentados abaixo:

```
4
5       # Criação da Tabela Clientes, com a chave primár
6 •     CREATE TABLE clientes (
7           id_cliente INT AUTO_INCREMENT PRIMARY KEY, -
8           nome VARCHAR(100) -- nome do cliente
9       );
```

```
# Criação da Tabela Equipamentos, com a chave primária sendo
CREATE TABLE equipamentos (
    id_equipamento INT AUTO_INCREMENT PRIMARY KEY, -- id do e
    id_cliente INT, -- foreign key que referencia ao cliente
    descricao VARCHAR(255), -- descrição do equipamento
    modelo VARCHAR(100), -- modelo do equipamento
    FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente)
);
```

```
# Criação da Tabela Orçamentos, para armazenar cada orçamento
CREATE TABLE orcamentos (
    id_orcamento INT AUTO_INCREMENT PRIMARY KEY, -- id do orç
    id_cliente INT, -- foreign key para o cliente que fez o or
    data_criacao DATETIME DEFAULT CURRENT_TIMESTAMP, -- data
    prazo_estimado VARCHAR(100), -- prazo estimado para entreg
    status ENUM('pendente', 'aprovado', 'recusado') DEFAULT 'p
    observacoes TEXT, -- campo para observações adicionais
    FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente)
);
```

```
# Criação da Tabela Serviços, par
CREATE TABLE servicos (
    id_servico INT AUTO_INCREMENT
    descricao TEXT, -- descrição
    mao_obra DECIMAL(10,2), -- va
    valor_pecas DECIMAL(10,2) --
);
```

```sql
# Tabela intermediária para associar um orçamento com serviços e equipame
CREATE TABLE orcamento_servicos (
    id_orcamento INT, -- foreign key para orçamento
    id_servico INT, -- foreign key para serviço
    id_equipamento INT, -- foreign key para equipamento
    PRIMARY KEY (id_orcamento, id_servico, id_equipamento), -- chave compo
    FOREIGN KEY (id_orcamento) REFERENCES orcamentos(id_orcamento),
    FOREIGN KEY (id_servico) REFERENCES servicos(id_servico),
    FOREIGN KEY (id_equipamento) REFERENCES equipamentos(id_equipamento)
);
```

Descrição das Tabelas

- **Tabela clientes**: Armazena informações dos clientes cadastrados no site.
  - cliente_id: Identificador único do cliente.
  - nome: Nome completo do cliente.
- **Tabela equipamentos**: Armazena informações sobre os equipamentos (ou produtos) que os clientes possuem ou desejam reparar. Neste contexto, "equipamentos" pode ser adaptado para qualquer produto ou item relacionado à loja.
  - id_equipamento: Identificador único do equipamento (chave primária).
  - id_cliente: Identificador do cliente que possui o equipamento (chave estrangeira, refere-se ao id_clientes da tabela clientes).
  - descricao: Descrição do equipamento (exemplo: tipo ou características do produto).
  - modelo: Modelo do equipamento (exemplo: versão ou marca).
- **Tabela orcamentos**: Registra os orçamentos realizados para os clientes.
  - id_orcamento: Identificador único do orçamento (chave primária).
  - id_cliente: Identificador do cliente que possui o equipamento (chave estrangeira, refere-se ao id_clientes da tabela clientes).
  - data_criacao: Data em que o orçamento foi criado
  - prazo_estimado: Prazo estimado para a entrega do orçamento ou serviço (campo de texto para flexibilidade).
  - status: Status do orçamento (exemplo: "Pendente", "Aprovado", "Recusado").
  - observacoes: Observações adicionais sobre o orçamento.
- **Tabela servicos**: Armazena os serviços que podem ser oferecidos aos clientes.
  - id_servico: Identificador único do serviço (chave primária).

- descricao: Descrição detalhada do serviço prestado (exemplo: manutenção, reparo, etc.).
- mao_obra: Valor da mão de obra para realizar o serviço.
- valor_pecas: Valor das peças utilizadas no serviço.

- **Tabela orcamento_servicos***:* Relaciona os orçamentos com os serviços e equipamentos envolvidos no orçamento.
  - id_orcamento: Identificador do orçamento (chave estrangeira, refere-se ao id_orcamento da tabela orcamentos)
  - id_servico: Identificador do serviço (chave estrangeira, refere-se ao id_servico da tabela servicos)
  - id_equipamento: Identificador do equipamento relacionado ao orçamento (chave estrangeira, refere-se ao id_equipamento da tabela equipamentos
  - PRIMARY KEY: Chave composta para evitar duplicação (envolve id_orcamento, id_servico e id_equipamento)

Consultas e Funcionalidades

VIEWS

1. **Lista de orçamentos com nome do cliente e status**

```sql
CREATE VIEW vw_orcamentos_clientes AS
SELECT o.id_orcamento, c.nome AS cliente, o.data_criacao, o.status
FROM orcamentos o
JOIN clientes c ON o.id_cliente = c.cliente_id;
```

2. **Equipamentos por cliente**

```sql
CREATE VIEW vw_equipamentos_por_cliente AS
SELECT c.nome, e.descricao, e.modelo
FROM equipamentos e
JOIN clientes c ON e.id_cliente = c.cliente_id;
```

3. **Detalhes de serviços por orçamento**

```sql
CREATE VIEW vw_detalhes_orcamento_servicos AS
SELECT o.id_orcamento, s.descricao, s.mao_obra, s.valor_pecas, (s.mao_obra + s.valor_pecas) AS total
FROM orcamento_servicos os
JOIN servicos s ON os.id_servico = s.id_servico
JOIN orcamentos o ON os.id_orcamento = o.id_orcamento;
```

4. **Orçamentos aprovados**

```sql
CREATE VIEW vw_orcamentos_aprovados AS
SELECT o.id_orcamento, c.nome, o.prazo_estimado
FROM orcamentos o
JOIN clientes c ON o.id_cliente = c.cliente_id
WHERE o.status = 'Aprovado';
```

Procedures

1. **Inserir novo cliente**

```sql
DELIMITER //
CREATE PROCEDURE sp_inserir_cliente(IN nome_cliente VARCHAR(100))
BEGIN
  INSERT INTO clientes (nome) VALUES (nome_cliente);
END //
DELIMITER ;
```

2. **Atualizar status de orçamento**

```sql
DELIMITER //
CREATE PROCEDURE sp_atualizar_status(IN id INT, IN novo_status VARCHAR(50))
BEGIN
  UPDATE orcamentos SET status = novo_status WHERE id_orcamento = id;
END //
DELIMITER ;
```

Functions

1. **Calcular custo total de um serviço**

```sql
DELIMITER //
CREATE FUNCTION fn_total_servico(id_serv INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
  DECLARE total DECIMAL(10,2);
  SELECT (mao_obra + valor_pecas) INTO total
  FROM servicos WHERE id_servico = id_serv;
  RETURN total;
END //
DELIMITER ;
```

2. **Contar orçamentos por cliente**

```
DELIMITER //
CREATE FUNCTION fn_orcamentos_por_cliente(id_cli INT)
RETURNS INT
DETERMINISTIC
BEGIN
  DECLARE qtd INT;
  SELECT COUNT(*) INTO qtd FROM orcamentos WHERE id_cliente = id_cli;
  RETURN qtd;
END //
DELIMITER ;
```

Triggers

1. **Atualiza status para “pendente” ao inserir novo orçamento**

```
DELIMITER //
CREATE TRIGGER trg_default_status
BEFORE INSERT ON orcamentos
FOR EACH ROW
BEGIN
  SET NEW.status = IFNULL(NEW.status, 'Pendente');
END //
DELIMITER ;
```

2. **Define automaticamente o prazo estimado como "5 dias úteis" quando o status do orçamento for alterado para "Aprovado" e o campo estiver vazio.**

```
DELIMITER //
CREATE TRIGGER trg_prazo_padrao_aprovado
BEFORE UPDATE ON orcamentos
FOR EACH ROW
BEGIN
  IF NEW.status = 'Aprovado' AND (NEW.prazo_estimado IS NULL OR NEW.prazo_estimado = '') THEN
    SET NEW.prazo_estimado = '5 dias úteis';
  END IF;
END //
DELIMITER ;
```

Transaction

5. **Transação para inserir um novo orçamento junto com os serviços relacionados, garantindo que ambos sejam gravados ou nenhum deles seja aplicado em caso de erro.**

```sql
-- Transação para inserir um novo orçamento junto com os serviços relacionados,
-- garantindo que ambos sejam gravados ou nenhum deles seja aplicado em caso de erro.
START TRANSACTION;

-- Inserir novo orçamento
INSERT INTO orcamentos (id_cliente, data_criacao, prazo_estimado, status, observacoes)
VALUES (1, NOW(), '7 dias', 'Pendente', 'Orçamento gerado via sistema');

-- Suponha que o ID do orçamento gerado foi 10
-- Inserir serviços associados ao orçamento
INSERT INTO orcamento_servicos (id_orcamento, id_servico, id_equipamento)
VALUES (10, 2, 3), (10, 4, 3);

COMMIT;
```
