Módulo Financeiro: Contas a Pagar e Receber

Kaique Xavier e Matheus Assis

Novembro de 2024

Resumo

Este documento apresenta o desenvolvimento de um módulo financeiro abrangente para a gestão de contas a pagar e a receber. O sistema proposto inclui funcionalidades como cadastro de clientes, registro de transações financeiras, geração de relatórios detalhados e acompanhamento do fluxo financeiro da organização. O objetivo é fornecer uma solução eficiente, escalável e de fácil utilização.

Introdução

A gestão financeira é um aspecto crucial para o sucesso de qualquer organização. Este projeto visa criar um módulo financeiro que simplifica o gerenciamento de contas a pagar e receber, garantindo precisão e agilidade nas operações financeiras. O módulo integra ferramentas de análise, relatórios gerenciais e controles internos, promovendo a tomada de decisões estratégicas.

Objetivos

O sistema foi projetado para atender os seguintes objetivos principais:

- Facilitar o cadastro e gerenciamento de clientes e fornecedores;
- Registrar e acompanhar contas a pagar e a receber;
- Oferecer relatórios financeiros detalhados, como:
  - Contas a vencer;
  - Contas vencidas;
  - Contas quitadas (pagas/recebidas).

- Automatizar cálculos de juros, multas e descontos com base em políticas definidas.
- Garantir segurança no acesso às informações financeiras.

Modelagem do Banco de Dados

A modelagem do banco de dados é fundamental para garantir a consistência e a eficiência das operações. O Diagrama Entidade-Relacionamento (DER) abaixo ilustra as principais entidades e seus relacionamentos:

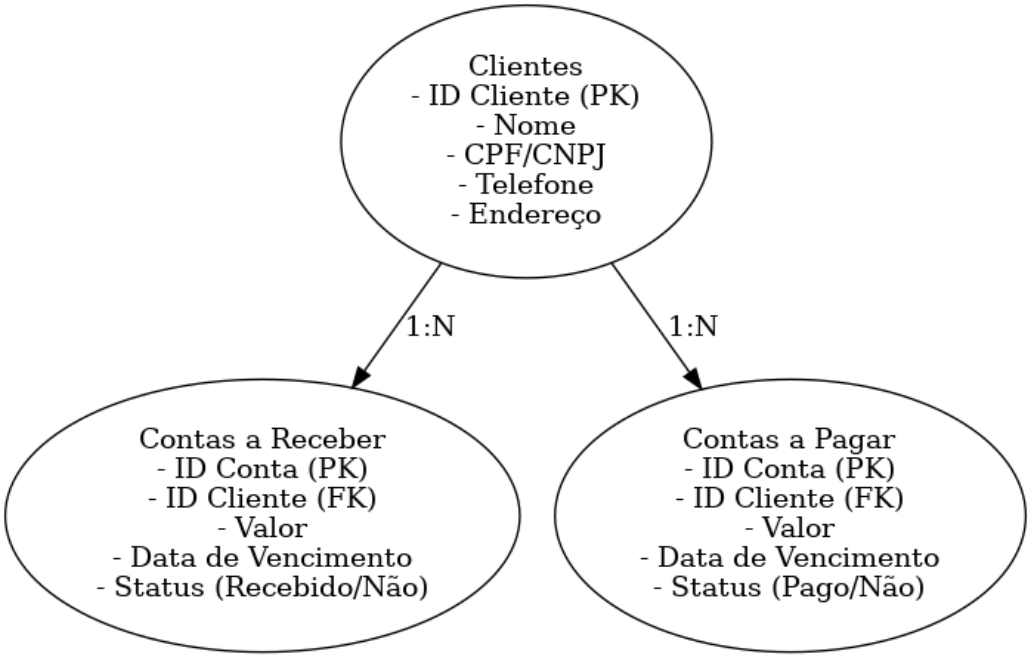


Figura 1: Diagrama Entidade-Relacionamento (DER) do Sistema

As principais entidades incluem:

- **Cliente:** armazena informações sobre clientes e fornecedores;
- **ContaFinanceira:** registra dados de contas a pagar e receber;
- **Transação:** representa movimentações financeiras associadas a contas;
- **Usuário:** gerencia acessos e permissões no sistema.

Fluxo de Atividades

O diagrama a seguir apresenta o fluxo principal das atividades realizadas no sistema, desde o cadastro de clientes até a análise de relatórios financeiros:

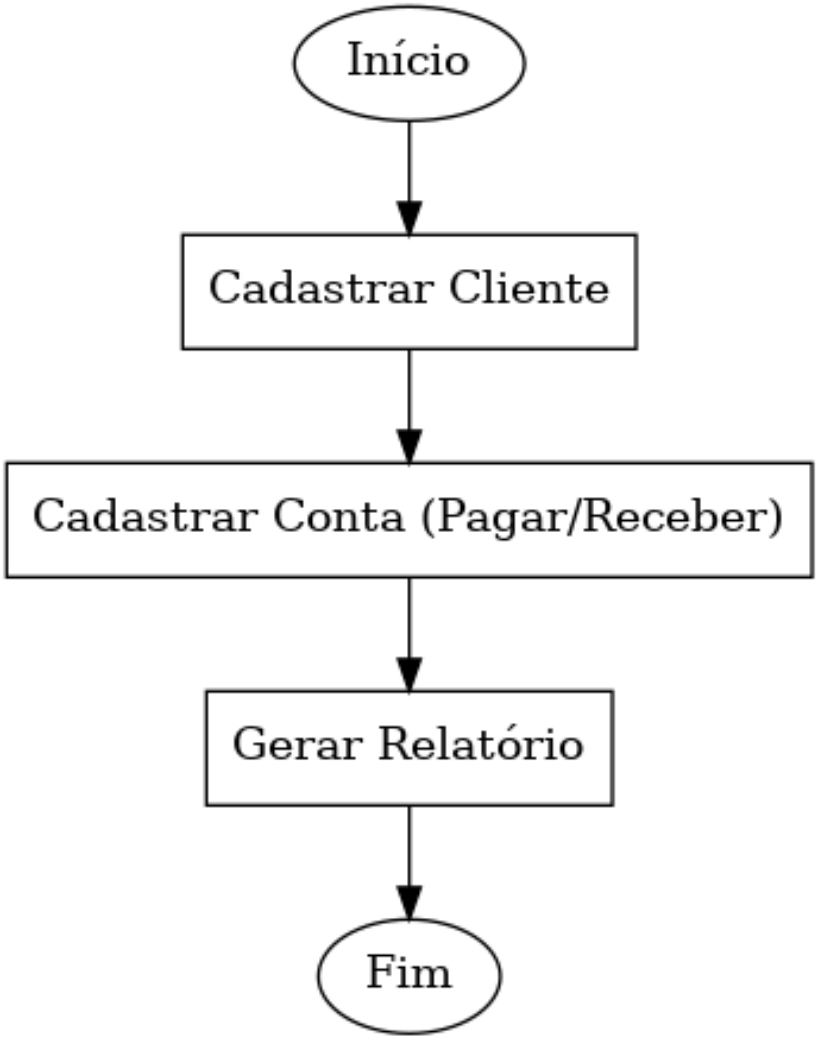


Figura 2: Diagrama de Atividades do Sistema

Diagrama de Classes

O Diagrama de Classes detalha a estrutura do sistema, com as principais classes e seus atributos, bem como os relacionamentos entre elas:

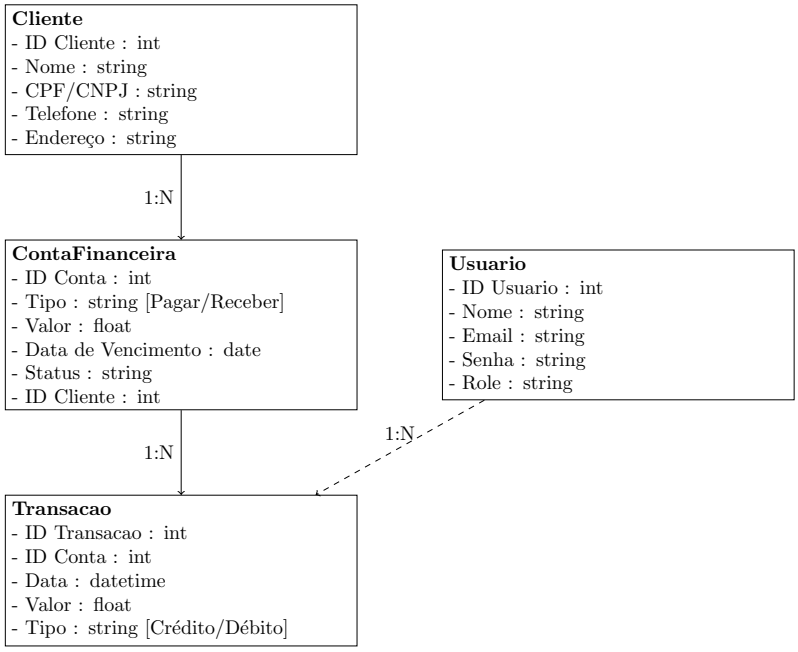


Figura 3: Diagrama de Classes do Sistema

Implementação

A implementação do módulo financeiro utiliza tecnologias modernas e boas práticas de desenvolvimento. Os principais aspectos incluem:

Conexão ao Banco de Dados

A conexão é realizada utilizando drivers otimizados para PostgreSQL e MySQL, com suporte a transações para garantir consistência em operações críticas.

Transações e Stored Procedures

As stored procedures são utilizadas para automatizar operações como cálculo de juros e aplicação de descontos em contas atrasadas. Exemplo de stored procedure para cálculo de juros:

```
CREATE PROCEDURE calcularJuros (IN id_conta INT)
BEGIN
    -- L g i c a  para  calcular  e  aplicar  juros  em  contas
       v e n c i d a s
END;
```

Exemplos de Consultas SQL

Exemplos de operações SQL frequentes que são utilizados em diferentes partes do sistema:

- **Dashboard:** No painel principal (dashboard), onde são exibidos os totais de clientes e o valor das contas a pagar e a receber, são feitas as seguintes consultas SQL:

```
-- Consulta para contar o n mero de clientes
   cadastrados
SELECT COUNT(*) FROM Cliente;

-- Consulta para calcular o total de contas a pagar
SELECT SUM(valor) AS total FROM ContaFinanceira WHERE
    tipo = 'pagar' AND status != 'paga';

-- Consulta para calcular o total de contas a receber
SELECT SUM(valor) AS total FROM ContaFinanceira WHERE
    tipo = 'receber' AND status != 'recebida';
```

- **Cadastro de Clientes:** Para o cadastro de clientes, as consultas SQL utilizadas para inserir e atualizar registros no banco de dados são:

```
-- Inser  o de novo cliente
INSERT INTO Cliente (nome, cpf, endereco, telefone)
   VALUES (?, ?, ?, ?);

-- Atualiza  o de dados de cliente
UPDATE Cliente SET nome = ?, endereco = ?, telefone =
    ? WHERE id = ?;
```

- **Relatórios:** A geração de relatórios financeiros é realizada por meio de consultas SQL que agrupam e filtram dados com base em parâmetros fornecidos pelos usuários. Alguns exemplos são:

```
-- Relat rio de contas a vencer nos pr ximos 30
   dias
SELECT * FROM ContaFinanceira WHERE data_vencimento
   BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL
    30 DAY);

-- Relat rio de contas vencidas
SELECT * FROM ContaFinanceira WHERE data_vencimento <
    CURDATE() AND status != 'paga';

-- Relat rio de contas pagas ou recebidas
SELECT * FROM ContaFinanceira WHERE status = 'paga'
   OR status = 'recebida';
```

Geração de Relatórios

A geração de relatórios é um dos pontos-chave do sistema, permitindo exportação para formatos como PDF e Excel. Exemplos de relatórios disponíveis:

- Relatório de contas a vencer nos próximos 30 dias;
- Análise de inadimplência por cliente/fornecedor;
- Histórico financeiro detalhado.

Consultas SQL Realizadas

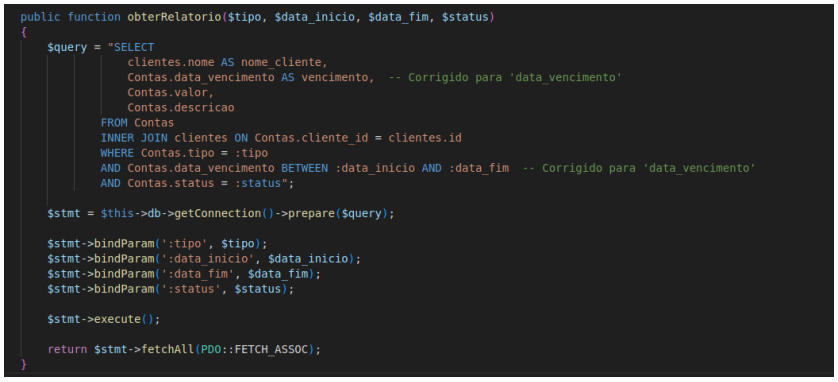
Diversas consultas SQL são executadas no sistema para realizar a coleta e análise de dados financeiros. Essas consultas são responsáveis por gerar as informações necessárias para o dashboard, os relatórios e os cálculos financeiros.

```php
public function contarClientes()
{
    $query = "SELECT COUNT(*) AS total FROM clientes";
    $stmt = $this->db->getConnection()->query($query);
    $resultado = $stmt->fetch(PDO::FETCH_ASSOC);
    return $resultado['total'] ?? 0;  // Retorna 0 caso não haja resultados
}
```

Figura 4: Consulta de total de clientes cadastrados no banco de dados.

```php
public function obterTotalContas($tipo)
{
    $query = "SELECT SUM(valor) AS total FROM Contas WHERE tipo = :tipo AND status != 'paga'"; // Exclui contas pagas
    $stmt = $this->db->getConnection()->prepare($query);
    $stmt->bindParam(':tipo', $tipo);
    $stmt->execute();
    $resultado = $stmt->fetch(PDO::FETCH_ASSOC);
    return $resultado['total'] ?? 0;  // Retorna 0 caso não haja resultados
}
```

Figura 5: Consulta para calcular o total de contas a pagar.



```php
public function obterRelatorio($tipo, $data_inicio, $data_fim, $status)
{
    $query = "SELECT
                clientes.nome AS nome_cliente,
                Contas.data_vencimento AS vencimento,  -- Corrigido para 'data_vencimento'
                Contas.valor,
                Contas.descricao
            FROM Contas
            INNER JOIN clientes ON Contas.cliente_id = clientes.id
            WHERE Contas.tipo = :tipo
            AND Contas.data_vencimento BETWEEN :data_inicio AND :data_fim  -- Corrigido para 'data_vencimento'
            AND Contas.status = :status";

    $stmt = $this->db->getConnection()->prepare($query);

    $stmt->bindParam(':tipo', $tipo);
    $stmt->bindParam(':data_inicio', $data_inicio);
    $stmt->bindParam(':data_fim', $data_fim);
    $stmt->bindParam(':status', $status);

    $stmt->execute();

    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

Figura 6: Consulta para gerar o relatório de contas vencidas.

Segurança

O sistema implementa medidas de segurança para proteger os dados financeiros, incluindo:

- Criptografia de dados sensíveis;
- Controle de acesso baseado em papéis;
- Auditoria completa das transações realizadas.

Conclusão

O módulo financeiro desenvolvido atende às necessidades de controle e gestão financeira de pequenas e médias empresas. Ele é flexível para personalização e expansão futura, garantindo uma base sólida para o crescimento organizacional. O próximo passo envolve testes abrangentes e coleta de feedback dos usuários finais.