Especificação Simples do Sistema Financeiro: Gerenciamento de Usuários e Contas

1. Introdução:

O Sistema Financeiro de Gerenciamento de Usuários e Contas tem como objetivo fornecer uma plataforma para que os usuários possam registrar-se, autenticar-se e gerenciar suas contas bancárias. A aplicação permitirá que os usuários realizem operações financeiras básicas, como visualização de saldo, transferência entre contas e histórico de transações.

2. Funcionalidades:

a.Cadastro de Usuário:

- Os usuários podem se registrar no sistema fornecendo informações como nome, e-mail e senha.

- As informações do usuário serão armazenadas em um banco de dados seguro.

b.Autenticação:

- Usuários registrados podem fazer login na plataforma usando seu e-mail e senha.

c.Gerenciamento de Contas Bancárias:

- Cada usuário terá uma conta bancária associada.

- O sistema permitirá que os usuários visualizem o saldo atual de sua conta.

d.Transferência de Fundos:

- Usuários podem transferir fundos entre suas contas ou para contas de outros usuários.

- O sistema validará se o usuário possui fundos suficientes antes de realizar a transferência.

e.Histórico de Transações:

- O sistema manterá um registro de todas as transações realizadas pelos usuários.

- Os usuários podem visualizar seu histórico de transações.

f.Segurança:

- As senhas dos usuários serão armazenadas de forma segura usando técnicas de hash.

- Todas as transações serão registradas para auditoria.

3. Estrutura do Banco de Dados:

a.Tabela de Usuários:

sql CREATE TABLE usuarios ( id\_usuario INT PRIMARY KEY, nome VARCHAR(50) NOT NULL, email VARCHAR(100) NOT NULL UNIQUE, senha VARCHAR(255) NOT NULL );

b.Tabela de Contas Bancárias:

sql CREATE TABLE contas\_bancarias ( id\_conta INT PRIMARY KEY, id\_usuario INT, saldo DECIMAL(10,2) DEFAULT 0, FOREIGN KEY (id\_usuario) REFERENCES usuarios(id\_usuario) );

c.Tabela de Transações:

sql CREATE TABLE transacoes ( id\_transacao INT PRIMARY KEY, id\_conta\_origem INT, id\_conta\_destino INT, valor DECIMAL(10,2) NOT NULL, data\_transacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, FOREIGN KEY (id\_conta\_origem) REFERENCES contas\_bancarias(id\_conta), FOREIGN KEY (id\_conta\_destino) REFERENCES contas\_bancarias(id\_conta) );

4. Considerações Finais:

Este sistema oferece uma estrutura básica para o gerenciamento de usuários e contas em um contexto financeiro. Para uma implementação completa, outros aspectos, como segurança adicional, otimizações de desempenho e recursos avançados, podem ser considerados e implementados. Este é um ponto de partida simples que pode ser expandido conforme necessário para atender aos requisitos específicos do projeto.

Especificação Técnica: Função de Validação de Dados Pessoais

Objetivo:

O objetivo desta especificação é definir os requisitos para o desenvolvimento de uma função que valide e-mails, CPFs, RGs e carteiras de motorista em uma aplicação.

Requisitos Funcionais:

Validação de E-mail:

A função deve verificar se o e-mail possui o formato correto.

Deve considerar a presença de "@" e pelo menos um ponto após o "@".

Exemplo válido: "usuario@dominio.com"

Validação de CPF:

A função deve verificar se o CPF possui 11 dígitos numéricos.

Deve realizar o cálculo do dígito verificador e verificar se o CPF é válido.

Exemplo válido: "123.456.789-09"

Validação de RG:

A função deve permitir diferentes formatos de RG (números e caracteres especiais).

Pode ser utilizado algum algoritmo de validação específico, dependendo do formato aceito.

Exemplo válido: "12.345.678-9" ou "MG-12.345.678"

Validação de Carteira de Motorista:

A função deve aceitar diferentes formatos de carteira de motorista.

Pode incluir validação de código específico para cada categoria.

Exemplo válido: "AB1234567890123"

Requisitos Não Funcionais:

Desempenho:

A função deve ter um desempenho aceitável para grandes volumes de validações.

O tempo de resposta deve ser razoável, mesmo em situações de carga elevada.

Manutenibilidade:

O código deve ser modular e fácil de entender.

Deve incluir comentários adequados para explicar a lógica de validação.

Segurança:

A função deve ser resistente a ataques de injeção de código ou manipulação maliciosa.

Documentação:

Deve ser fornecida documentação clara sobre como usar a função, incluindo exemplos de chamadas e formatos aceitáveis.

Observações:

A função deve retornar um valor booleano indicando se os dados são válidos ou não.

Mensagens de erro específicas devem ser retornadas em caso de invalidade, indicando o motivo.

Aceitação:

A função será considerada aceita quando os requisitos funcionais e não funcionais forem atendidos de acordo com a especificação.

Essa é uma especificação básica que pode ser ajustada conforme as necessidades específicas do seu projeto e da linguagem de programação que você está usando. Certifique-se de adaptar os detalhes conforme necessário.