Documentação do Sistema de Gestão Financeira

1. Visão Geral

Este sistema é um software de gestão financeira desenvolvido em C++ para ajudar usuários a gerenciar receitas, despesas, metas financeiras e investimentos. Ele possui integração com APIs financeiras para atualizar automaticamente os preços de ativos como ações, moedas e criptomoedas, oferecendo uma visão em tempo real dos investimentos.

2. Funcionalidades Principais

- **Gerenciamento de Receitas e Despesas**: Registro de entradas e saídas financeiras.
- **Definição de Metas Financeiras**: Criação e monitoramento de metas com prazo e valor.
- **Gestão de Investimentos**: Registro e atualização automática do valor de investimentos através de APIs financeiras.
- **Relatórios Financeiros**: Geração de relatórios detalhados e gráficos para análise do portfólio.

3. Tecnologias e Bibliotecas Utilizadas

3.1 Banco de Dados

- **SQLite**: Para armazenamento persistente de dados locais de receitas, despesas, metas e investimentos. Alternativas incluem MySQL e PostgreSQL, caso seja necessária uma infraestrutura de rede.

3.2 Requisições HTTP

- **libcurl**: Biblioteca para realizar requisições HTTP/HTTPS às APIs financeiras e buscar os dados de cotações e preços.

3.3 Manipulação de Dados JSON

- **nlohmann/json**: Biblioteca para parse e manipulação de dados JSON, utilizada para processar as respostas das APIs.

3.4 Manipulação de Datas e Horas

- **Chrono (Biblioteca padrão do C++)**: Utilizada para lidar com datas e calcular prazos de metas e registros financeiros.

3.5 Gráficos (Opcional)

- **SFML ou matplotlib-cpp**: Para geração de gráficos básicos que mostram o fluxo de caixa e a evolução dos investimentos (opcional).

3.6 Criptografia (Opcional)

- **OpenSSL**: Para segurança de dados confidenciais, como armazenamento seguro de senhas.

4. Arquitetura do Sistema

4.1 Estrutura do Banco de Dados

- **Tabela usuarios**: Armazena informações dos usuários.
- **Tabela receitas e despesas**: Guardam transações financeiras.
- **Tabela metas_financeiras**: Contém as metas criadas pelo usuário.
- **Tabela investimentos**: Armazena detalhes dos ativos e seu valor atual.

4.2 Estrutura das Classes

Classes principais:

- **Usuario**: Armazena informações pessoais e autenticação.
- **Transacao**: Classe base para receitas e despesas.
- **MetaFinanceira**: Armazena as metas do usuário.
- **Investimento**: Registra detalhes dos ativos e métodos para atualização de valores.

5. Integração com APIs de Dados Financeiros

Para obter os valores atualizados dos investimentos, utilizamos APIs de mercado financeiro, como CoinGecko ou Alpha Vantage. Segue o processo básico de integração:

1. **Configuração e Requisição HTTP**
    - Utilizar libcurl para fazer requisições GET aos endpoints das APIs.
2. **Parse dos Dados JSON**
    - Receber os dados da API em JSON e processá-los com a biblioteca nlohmann/json.
3. **Atualização do Banco de Dados**
    - Após obter o valor atual do ativo, atualize o campo correspondente no banco de dados.

6. Considerações sobre Segurança

Para proteger dados confidenciais (ex., senhas), recomenda-se:

- **Hashing de Senhas**: Utilizar SHA-256 (disponível no OpenSSL) para armazenar hashes de senhas no banco de dados.
- **Armazenamento Seguro de Chaves de API**: Chaves de API para acessar dados financeiros devem ser armazenadas em arquivos de configuração fora do código-fonte.

7. Conclusão

Este sistema oferece uma solução robusta para o gerenciamento de finanças pessoais, com funcionalidades de monitoramento e atualização automática de investimentos via

APIs externas. Com uma interface modular, o sistema pode ser estendido para incluir gráficos, criptografia e outros recursos adicionais conforme necessário.

Esse resumo fornece uma base sólida para desenvolver e expandir o sistema conforme os requisitos específicos de funcionalidade e segurança.