Documento de Especificação de Requisitos

Introdução

Este documento tem como objetivo definir os requisitos funcionais e não funcionais do Sistema para gestão de gastos de uma residência em Python. O controle de gastos de uma residência é extremamente importante visto que existem diversas despesas dentro de um ambiente residencial, dessa forma, visando monitorar e controlar essas despesas foi criado o Sistema de gestão de gastos em Python. O sistema deve permitir que o usuário adicione, edite, exclua, liste despesas, e mostre o saldo atual com base nas despesas (gastos) já adicionadas, além de salvar os dados em um arquivo JSON para que os mesmos possam ser consultados novamente quando necessário.

História de Usuário 01:

Eu, como usuário, quero adicionar uma nova despesa, para manter o controle do meu orçamento.

Requisito Funcional 01 - Adicionar Transação

O sistema deve permitir que o usuário adicione uma nova despesa. Ela deve conter uma descrição, e um valor. Se o valor for negativo, a despesa será inserida e posteriormente exibida com sinal (-) demonstrando o gasto.

Critérios de Aceitação:

O sistema deve solicitar ao usuário a descrição, e o valor da despesa. A despesa deve ser armazenada em um dicionário que será salvo em um arquivo JSON.

Cenários de Testes de Aceitação:

O usuário adiciona uma nova despesa com valor negativo. A despesa deve ser adicionada ao dicionário e salva posteriormente para consulta edição ou exclusão da mesma.

História de Usuário 02:

Eu, como usuário, quero editar uma despesa existente para corrigir informações incorretas ou atualizar o registro

Requisito Funcional 02 - Editar Transação:

O sistema deve permitir que o usuário edite uma despesa existente. O usuário deve ser capaz de editar a descrição, o valor ou a data da transação.

Critérios de Aceitação:

O sistema deve solicitar ao usuário a descrição da despesa que deseja editar. Se a despesa for encontrada, o sistema deve permitir que o usuário edite a descrição, o valor ou a data. As alterações devem ser salvas no dicionário e mantidas no arquivo JSON.

Cenários de Testes de Aceitação:

O usuário tenta editar uma despesa que não existe na carteira. O sistema deve informar que ela não foi encontrada.

O usuário edita a descrição de uma despesa existente. As informações devem ser atualizadas no dicionário e no arquivo JSON.

História de Usuário 03:

Eu, como usuário, quero excluir uma despesa existente para remover um registro incorreto ou desnecessário.

Requisito Funcional 03 - Excluir Transação

O sistema deve permitir que o usuário exclua uma despesa existente. O usuário deve ser capaz de excluir uma despesa por sua descrição.

Critérios de Aceitação:

O sistema deve solicitar ao usuário a descrição da despesa que deseja excluir. Se a despesa for encontrada, o sistema deve exclui-la do dicionário e manter as alterações no arquivo JSON.

Cenários de Testes de Aceitação:

O usuário tenta excluir uma despesa que não existe na carteira. O sistema deve informar que ela não foi encontrada.

O usuário exclui uma despesa existente. A despesa deve ser removida do dicionário e as alterações devem ser persistidas no arquivo JSON.

História de Usuário 04:

Eu, como usuário, quero listar todas as despesas existentes para visualizar o meu histórico de despesas.

Requisito Funcional 04 - Listar Transações:

O sistema deve permitir que o usuário liste todas as despesas existentes na carteira.

Critérios de Aceitação:

O sistema deve exibir a lista de despesas com a descrição, o valor e a data de cada transação. As despesas devem ser exibidas em ordem cronológica inversa, ou seja, da mais recente para a mais antiga.

Cenários de Testes de Aceitação:

O usuário lista as despesas existentes na carteira. O sistema deve exibir a lista em ordem cronológica inversa, com a descrição, o valor e a data de cada transação.

História de Usuário 05:

Eu, como usuário, quero somar todas as despesas existentes para visualizar o gasto total.

Requisito Funcional 05 - Listar Transações:

O sistema deve permitir que o usuário some todas as despesas existentes na carteira.

Critérios de Aceitação:

O sistema deve somar a lista de despesas. A soma das mesmas deve ser exibida para o usuário após a seleção da opção desejada.

Cenários de Testes de Aceitação:

O usuário soma as despesas existentes na carteira. O sistema deve exibir o resultado da soma de todas as despesas e exibir o resultado da soma para o usuário.

História de Usuário 06:

Eu, como usuário, quero calcular meu saldo atual com base nas despesas que eu tenho salvas na carteira de gastos.

Requisito Funcional 06 - Listar Transações:

O sistema deve permitir que o usuário insira o seu saldo atual, e calcule a diferença entre o saldo inserido e o total de despesas que estão presentes dentro da carteira de gastos.

Critérios de Aceitação:

O sistema deve pedir para o usuário que insira o saldo atual e calcule a diferença entre o saldo e as despesas da carteira, em seguida ele deve retornar o resultado do saldo atual com base nessa diferença.

Cenários de Testes de Aceitação:

O usuário insere o saldo atual. O sistema deve fazer o cálculo da diferença entre o saldo atual e a soma das despesas presente na carteira de gastos.

Requisitos não Funcionais:

Confiabilidade: o código deve ser capaz de carregar e salvar informações da carteira de gastos a partir de um arquivo JSON, além de tratar exceções caso não consiga carregar as informações.

Disponibilidade: a classe Carteira deve ser implementada com o padrão de projeto Singleton, garantindo que apenas uma instância da classe seja criada e esteja disponível para uso em toda a aplicação.

Manutenibilidade: o código deve ser organizado de maneira clara e concisa, utilizando comentários para descrever a funcionalidade de cada método e explicando o padrão de projeto utilizado na classe Carteira.

Usabilidade: o código deve fornecer uma interface de linha de comando para o usuário, permitindo que ele adicione, liste, edite, exclua, some, e calcule o saldo atual de maneira intuitiva e eficiente.

Conclusão:

Este documento especificou os requisitos funcionais e não funcionais do Sistema de controle de gastos residenciais em Python. O sistema deve permitir que o usuário adicione, edite, exclua e liste despesas. Com a implementação desses requisitos, o sistema fornecerá uma solução simples e eficaz para gerenciar o orçamento residencial.