UNIVERSIDADE DE FORTALEZA
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: PROGRAMAÇÃO FUNCIONAL

TRABALHO FINAL - CAMPEONATO

INTEGRANTES DA EQUIPE:
ADILSON JÚNIOR VIEIRA RODRIGUES - MATRÍCULA: 2222805
LARA FERNANDA BEZERRA SAMPAIO - MATRÍCULA: 2222769
LEONARDO DE FREITAS RABELO - MATRÍCULA: 2323736

FORTALEZA-CE
Setembro/2024

1. Introdução

Este documento descreve as especificações do sistema de pontuações e penalidades. São apresentados os requisitos funcionais e não funcionais que orientam a implementação do sistema. Os requisitos funcionais detalham as funcionalidades do sistema, como a filtragem e ordenação de usuários com base em pontuações, aplicação de penalidades e a identificação de campeões. Já os requisitos não funcionais cobrem aspectos do desempenho, validação de entradas e feedback ao usuário.

2. Definição dos papéis

Cada membro da equipe teve uma função específica na construção do sistema, contribuindo para diferentes aspectos do desenvolvimento e da documentação.

Leonardo de Freitas Rabelo: Implementação do código;
Adilson Júnior Vieira Rodrigues: Casos de testes;
Lara Fernanda Bezerra Sampaio: Elaboração do documento de requisitos.

3. Requisitos funcionais

[RF01] Uso de lambda function: O sistema deve permitir que o usuário insira um valor de score mínimo para aprovação, maior que 0 e menor que 100, para filtrar os usuários aprovados.
Implementação: função **filterUsersByScore,** chamada na **main.**

[RF02] Uso de lambda function: O sistema deve filtrar usuários com base em um score mínimo.
Implementação: função **filterUsersByScore.**

[RF03] Uso de list comprehension: O sistema deve atualizar os scores dos usuários, adicionando pontos extras, sem ultrapassar o valor máximo de 100 e gerar uma nova lista com nome, score e grupo de trabalho.
Implementação: função **usersWithExtraPoints,** chamada na **main.**

[RF04] Uso de closure: O sistema deve calcular o score total de cada grupo de trabalho, considerando todos os membros do grupo.
Implementação: função **totalTeamScore.**

[RF05]Uso de função de alta ordem: O sistema deve aplicar uma redução de score aos usuários de um grupo de trabalho informado.
Implementação: função **applyReduction** e **reduction,** chamadas em **getUsersComPenalidades.**

[RF06] O sistema deve exibir os usuários ordenados com base na pontuação final após as pontuações extras e penalidades.
Implementação: função **sortedUsersByScore,** chamada na **main.**

[RF07] O sistema deve identificar qual grupo obteve a maior pontuação e anunciar o vencedor.

Implementação: Na seção “Resultados finais” o **totalTeamScore** está sendo usado para calcular o grupo vencedor na **main.**

[RF08] O sistema deve identificar o usuário com a maior pontuação após a aplicação da pontuação extra e penalidades, e defini-lo como o campeão individual.
Implementação: O campeão individual é o primeiro usuário da lista **usuariosFiltradosEOrdenados,** obtido através da função **sortedUsersByScore,** chamada na **main.**

4. Requisitos não funcionais

[RNF01] O sistema deve permitir entradas do usuário, como a inserção de scores mínimos e penalidades.
Implementação: Uso de **input()** para coletar dados de entrada do usuário**.**

[RNF02] O sistema deve garantir a validação das entradas, verificando se os valores fornecidos estão no intervalo correto (maior que 0 e menor que 100) ou se são válidos.
Implementação: try-except para captura de exceções ao usar **input()** para **minScoreInput** e a função **getUsersComPenalidades.**

[RNF03] O sistema deve garantir que os objetos de usuários não sejam alterados diretamente durante as operações, mantendo uma cópia modificada.
Implementação: Uso do operador ****** na função **reduction.**

[RNF04] O sistema deve fornecer uma interface de feedback por meio do console, exibindo informações sobre pontuações parciais, totais e resultados de cada operação.
Implementação: Uso de **print()** para exibir as etapas dos usuários, como a pontuação parcial, pontuação extra, penalidades aplicadas, pontuação final, vencedor do grupo e campeão individual.

[RNF05] O sistema deve ser fácil de manter, com código modular, facilitando alterações ou correções futuras.
Implementação: Uso de funções separadas para diferentes funcionalidades **(filterUsersByScore, applyReduction, totalTeamScore),** garantindo modularidade.