UNIVERSIDADE DE FORTALEZA

DISCIPLINA: Programação Funcional

Ranking de Estudantes

GITHUB: github.com/pliniogoncalves/SistemaNotas

Testes Trello: https://trello.com/b/yQTuNv41/atividade-programacao-funcional-unifor

INTEGRANTES DA EQUIPE:

Artur Silva Barbosa Nunes - 2318047

Riana Brigida Pereira dos Santos - 2313683

Viviane Batista Brindeiro - 2317918

João Alexandre Bezerra da Silva - 2326291

Plínio Ricardo Gonçalves - 2313621

2025

Ranking de Estudantes

1. Objetivo do Sistema

A aplicação organiza o ranking dos estudantes, calcula as médias, permite adicionar e remover alunos, além de exibir suas notas e classificações através de uma interface GUI. O ranking é atualizado de forma automática, destacando os alunos com melhor desempenho e aqueles de recuperação.

2. Rodar o Projeto

Execute o comando no terminal:

- **python SistemaNotas.py**

Link do projeto no GitHub:

3. Requisitos Funcionais

- **RF01: Adicionar Estudante**

  Permite cadastrar estudantes e suas notas. Válida se as notas estão entre 1 e 10.

- **RF02: Remover Estudante**

  Permite excluir um estudante do ranking ao selecioná-lo.

- **RF03: Calcular e Exibir a Média**

  Calcula a média das notas de cada estudante e as exibe no ranking.

- **RF04: Exibir Ranking de Estudantes**

  Ordena os estudantes pela média, do maior para o menor.

- **RF05: Exibir Melhores Estudantes**

Mostra quais estudantes tiveram média igual ou superior a 7.

- **RF06: Exibir Estudantes em Recuperação**

  Indica os estudantes com média entre 5 e 6,9.

- **RF07: Exibir Estudantes Reprovados**

  Indica os estudantes com média abaixo de 5.

4. Requisitos Não Funcionais

- **RNF01: Interface Gráfica**

  O sistema possui uma interface desenvolvida com a biblioteca Tkinter.

- **RNF02: Validação de Entrada**

  Verifica se os dados inseridos são válidos e exibe mensagens de erro.

- **RNF03: Performance**

  O sistema atualiza o ranking, melhores estudantes, estudantes em recuperação e estudantes reprovados em tempo real.

- **RNF04: Facilidade de Uso**

  Campos de entrada do usuário e botões para adicionar, remover e visualizar os estudantes.

5. Estrutura do Código

5.1 Funções Principais

- **validar_entrada(nome, notas):**
  Função de validação que garante que as entradas sejam válidas. O nome não pode ser vazio, não pode haver numeros, ter menos de duas palavras com 3 letras em cada. As notas precisam ser numéricas, e todas as notas devem estar dentro do intervalo de 1 a 10. Ao inserir as notas, use o ponto (.) em vez da vírgula (,).

- **calcular_media(notas):**
  Função lambda que calcula a média das notas do estudante.

- **adicionar_estudante_func():**
  Função de alta ordem que gerencia o fluxo de adicionar um estudante ao ranking. Ela valida as entradas, calcula a média e atualiza o ranking.

- **remover_estudante():**
  Permite remover um estudante selecionado no ranking.

- **atualizar_ranking():**
  Ordena os estudantes com base na média das notas e exibe o ranking atualizado. Atualiza as listas de Melhores Estudantes e Estudantes em Recuperação e estudantes reprovados.

- **limpar_campos():**
  Limpa os campos de entrada após a adição de um estudante.

5.2 Conceitos de Programação Funcional Utilizados

- **Funções Lambda**: Usadas em `calcular_media` e para ordenar estudantes no ranking.

- **List Comprehension**: `estudante["media"] >= 7` Utilizadas para filtrar melhores estudantes e estudantes em recuperação.

- **Closure**: A função `adicionar_estudante_func` é um closure, com acesso às variáveis do escopo em que foi criada.

- **Funções de Alta Ordem**: `criar_validador` retorna uma função personalizada de validação.

6. Mapeamento de Requisitos e Código

Requisito Funcionais	Função Implementada
RF01: Adicionar Estudante	adicionar_estudante_func()
RF02: Remover Estudante	remover_estudante()
RF03: Calcular e Exibir a Média	calcular_media(notas)

RF04: Exibir Ranking de Estudantes	estudante["media"] >= 7
RF05: Exibir Melhores Estudantes	melhores_estudantes = [estudante for estudante in self.estudantes if estudante["media"] >= 7]
RF06: Exibir Estudantes em recuperação	recuperacao = [estudante for estudante in self.estudantes if 5 <= estudante["media"] < 7]
RF07: Exibir Estudantes Reprovados	reprovados = [estudante for estudante in self.estudantes if estudante["media"] < 5]
Requisitos Não Funcionais	**Função Implementada**
RNF01: Interface Gráfica	Tkinter
RNF02: Validação de Entrada	validar_entrada() e criar_validador()
RNF03: Performance	Atualização do ranking em tempo real
RNF04: Facilidade de Uso	Design intuitivo com botões e listbox

7. Interface Gráfica (GUI)

Seções da Interface:

1. **Entrada de Dados**: Campos para inserir nome e três notas do estudante.
2. **Botões**:
    - **Adicionar**: Adiciona um estudante.
    - **Remover**: Remove o estudante selecionado.
3. **Ranking**:
    - Listbox que exibe os estudantes ordenados pela média.
4. **Exibição**:

    Melhores Estudantes: Média ≥ 7.

    Recuperação: Média entre 5 e 6,9.

    Reprovados: Média < 5.

8. Casos de Teste

8.1 Adicionar Estudante

- **Entrada Válida**:
    - Inserir nome e notas válidas deve adicionar o estudante ao ranking.
- **Notas não numéricas**:
    - Exibir erro indicando que as notas devem ser numéricas.
- **Notas fora do intervalo**:
    - Exibir erro indicando que as notas devem estar entre 1 e 10.

8.2 Remover Estudante

- **Remoção Válida**:
    - O estudante selecionado é removido do ranking.
- **Remoção sem seleção**:
    - Exibir erro indicando que é necessário selecionar um estudante.

9. Responsabilidade da Equipe

Documentação: Riana Brigida

- Responsável por criar e organizar o documento de requisitos.
- Mapear os requisitos funcionais e não funcionais.
- Garantir que os requisitos estejam descritos no documento e no código.

Desenvolvimento da Interface Gráfica: Viviane Batista

- Implementar a interface gráfica usando Tkinter.
- Garantir que os requisitos sejam atendidos.
- A interface deve permitir fácil interação com o sistema como inserção de notas, adição e remoção de estudantes.

Implementação da Lógica Principal: Arthur, João, Plínio

- Implementação das funcionalidades, cálculo de médias, validação de entrada e atualização do ranking.
- Usando conceitos de programação funcional, como funções lambda, list comprehension, closures e funções de alta ordem.

Testes: João, Plínio

https://trello.com/b/yQTuNv41/atividade-programacao-funcional-unifor

- Criar casos de teste para validar a funcionalidade do sistema.
- Testar entradas válidas e inválidas, verificar atualizações em tempo real e garantir que os requisitos estejam atendidos.

Link:

GITHUB: github.com/pliniogoncalves/SistemaNotas