ENGENHARIA DE SOFTWARE
TRABALHO FINAL

Proposta: Sistema de Eventos de Robótica

Alunos: José Augusto Laube e Gustavo de Souza
Professora: Rebeca Schroeder Freitas

github: https://github.com/JoseLaube/SOFT

Descrição do Problema

Eventos de robótica, como competições de combate e seguidor de linha, frequentemente sofrem com a desorganização logística. O gerenciamento de equipes, a criação de chaves de batalha, o registro de tempos e a divulgação de resultados são, em muitos casos, realizados de forma manual, utilizando planilhas, documentos impressos e canais de comunicação dispersos. Este método é propenso a erros humanos, causa atrasos, gera falta de transparência nos resultados e resulta em uma experiência frustrante tanto para os competidores quanto para a equipe organizadora.

O **RoboComp Manager** surge como uma solução para este problema, propondo uma plataforma digital que centraliza e automatiza a gestão completa do evento, desde a inscrição das equipes até a divulgação dos resultados finais, garantindo agilidade, precisão e uma experiência aprimorada para todos os envolvidos.

Escopo do Software

- **Gestão de Usuários:** Permitir o cadastro e a autenticação de usuários com diferentes níveis de permissão (Organizador, Líder de Equipe, Juiz).

- **Gestão de Equipes e Robôs:** Permitir que Líderes de Equipe cadastrem suas equipes, gerenciem seus membros e registrem os robôs associados à equipe.

- **Gestão de Eventos:** Conceder ao Organizador a capacidade de criar novos eventos e definir as competições que o compõem (ex: Combate de Robôs 1kg, Seguidor de Linha Pro).

- **Módulo de Competição de Combate:**
    - Gerenciar a inscrição de robôs na categoria.
    - Gerar automaticamente chaves de batalha no formato de eliminatória simples.
    - Permitir que um Juiz registre o vencedor de cada luta.
    - Atualizar a chave de batalhas em tempo real conforme os resultados são inseridos.

- **Módulo de Competição de Seguidor de Linha:**
    - Gerenciar a inscrição de robôs na categoria.
    - Permitir que um Juiz registre múltiplas tomadas de tempo para cada robô.
    - Calcular e armazenar automaticamente o melhor tempo de cada competidor.
    - Gerar uma tabela de classificação ordenada pelo melhor tempo.

- **Painel de Visualização Pública:** Disponibilizar uma página de acesso público (sem necessidade de login) para exibir as chaves de batalha, horários, resultados e classificações

Stakeholders

Os stakeholders (partes interessadas) do projeto e seus principais interesses são:

- **Organizadores do Evento:** São os usuários primários do sistema. Necessitam de ferramentas para criar, configurar e gerenciar todas as fases do evento de forma eficiente e com controle total.
- **Líderes de Equipe:** Precisam de uma interface intuitiva para realizar o cadastro de suas equipes, membros e robôs, além de inscrevê-los nas competições desejadas.
- **Juízes/Árbitros:** Requerem uma interface simples e ágil para registrar os resultados das lutas e os tempos das corridas de forma rápida e precisa durante o evento.
- **Competidores (Membros da Equipe):** Têm interesse em visualizar informações cruciais como horários de suas lutas/corridas, a estrutura da chave do torneio e os resultados atualizados.
- **Público/Espectadores:** Desejam acompanhar o andamento da competição, visualizar as chaves, saber quem está vencendo e quais serão as próximas disputas.

Requisitos do Software

Os requisitos do sistema são especificados a seguir no formato de **Histórias de Usuário (User Stories)**, que descrevem as funcionalidades sob a perspectiva do valor gerado para cada tipo de usuário.

Perfil: Organizador

- **Como um** Organizador, **eu quero** criar um novo evento definindo seu nome e data, **para que** eu possa iniciar o planejamento e a divulgação de uma nova competição.
- **Como um** Organizador, **eu quero** visualizar e aprovar ou rejeitar as inscrições de equipes, **para que** eu possa garantir que apenas equipes que cumprem os pré-requisitos participem do evento.
- **Como um** Organizador, **eu quero** gerar as chaves de uma competição de combate com um único clique após o fim das inscrições, **para que** o processo seja rápido, imparcial e livre de erros manuais.

Perfil: Líder de Equipe

- **Como um** Líder de Equipe, **eu quero** me cadastrar e fazer login no sistema, **para que** eu possa ter acesso às funcionalidades de gerenciamento da minha equipe.
- **Como um** Líder de Equipe, **eu quero** cadastrar os membros da minha equipe com nome e função, **para que** haja um registro oficial de todos os competidores.
- **Como um** Líder de Equipe, **eu quero** inscrever um dos robôs da minha equipe em uma competição específica do evento, **para que** ele esteja oficialmente apto a competir.

Perfil: Juiz

- **Como um** Juiz de Combate, **eu quero** selecionar uma luta da chave e registrar o robô vencedor, **para que** o sistema atualize a chave automaticamente e defina o próximo confronto.

- **Como um** Juiz de Seguidor de Linha, **eu quero** registrar as três tomadas de tempo para um robô competidor, **para que** o sistema possa calcular e exibir seu melhor tempo na classificação geral.

Perfil: Competidor / Público

- **Como um** competidor, **eu quero** visualizar a chave de batalhas da minha competição em tempo real, **para que** eu saiba quem e quando será meu próximo oponente.
- **Como espectador**, **eu quero** acessar uma página pública para ver a tabela de classificação da competição de Seguidor de Linha, **para que** eu possa saber quem são os robôs mais rápidos do evento.

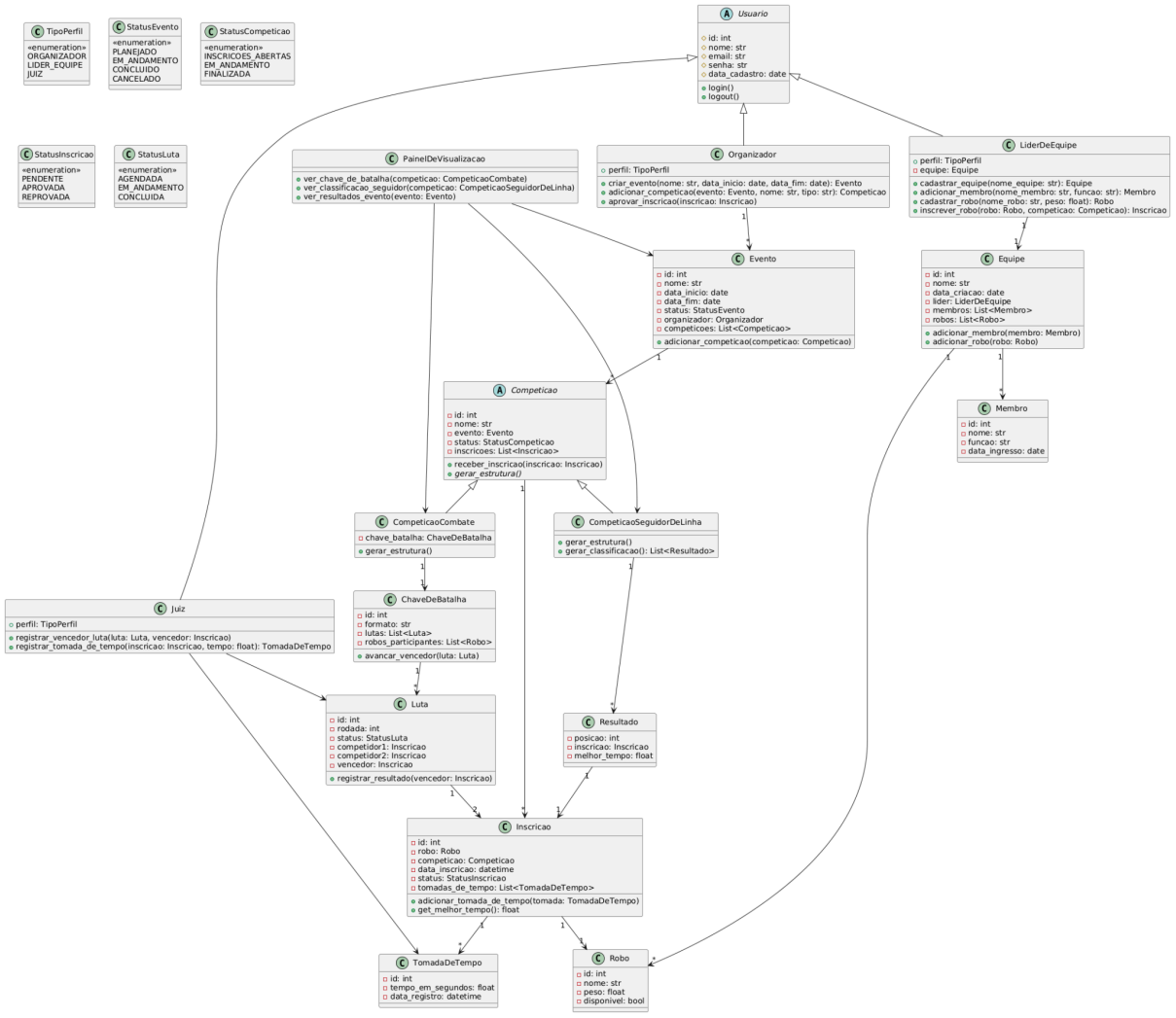
Tempo de duração do projeto (COCOMO)

Etapa	**Estimativa de linhas**
Modelos/Entidades	**400**
Gestão de usuários	**350**
Gestão de equipes/robôs	**400**
Gestão de eventos	**300**
Módulo Combate	**500**
Módulo seguidor de linha	**300**
interface do banco de dados	**250**
Painel de visualização	**450**
testes unitários	**400**

Total estimado (LOC) = 3350 == 3.35KLOC

Para esse projeto vamos considerar uma complexidade orgânica, e um nível básico, sendo assim temos:
Esforço = 2.4 * kLOC^(1.05) - 2.4 * 3.35^(1.05) - 8.54 (pessoas/mês)
Ducação = 2.5 * KLOC^(0.38) - 2.5 * 3.35^(0.38) - 3.48 (meses)

Diagrama UML:
Para fazer o diagrama UML usamos a ferramenta plantuml (https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa7000) que gera o diagrama a partir de um script, o script do nosso diagrama de classes está na pasta do github.
O diagrama de classes ficou da seguinte forma:



Classes Enums - representam tipo pré definidos;

Classe Usuário - abstrata - define os atributos e métodos básicos para os 3 tipos de usuários (organizador, líder da equipe e juiz) que tem acesso para modificar o conteúdo do sistema;

O líder de equipe, pode liderar uma equipe, que tem só um líder, e um ou mais membros, a equipe pode ter 1 ou mais robôs.

O organizador pode criar um ou mais eventos, um evento pode ter um ou mais competições podendo ser combate, ou seguidor de linha. A competição tem uma ou mais inscrição.

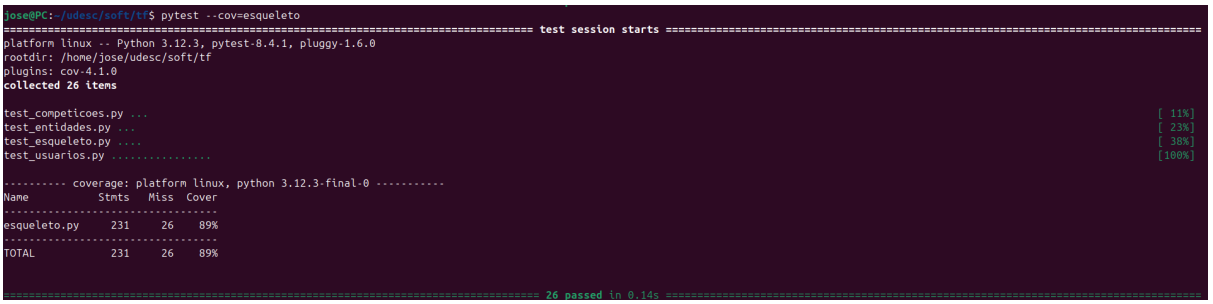
O juiz define uma luta e regista a tomada de tempo.

No painel de visualização pode ver o evento, e as competições, dentro da competição do seguidor de linha pode-se ver o resultados e na competição de combate pode ver as chaves e as lutas.

Testes unitários:

A base para o esqueleto do nosso código foi feita usando a biblioteca puml-tools de python que recebe o script gerado no plant uml e a partir dele gera o código python, demos só algumas incrementadas em partes que julgamos necessárias, para termos o esqueleto do nosso sistema.

Então foi possível definir os teste unitários para o nosso sistema, não conseguimos atingir uma cobertura de 100%, mas testamos as partes mais cruciais do nosso código.
Os nossos testes tem uma cobertura de 89% e foi usado a biblioteca pytest para fazer os testes, essa biblioteca gera um html do código mostrando que partes dele estão cobertas, esse html está no github, para maior análise.



As classes que não foram cobertas pelos nossos testes unitários estão relacionadas à lógica de visualização das chaves, e das tomadas de tempo, bem como os métodos que fazem a atualização dessas informações. Julgamos que os métodos relacionados aos usuários e eventos eram mais importantes para o nosso sistema, por isso focamos em realizar esses testes de cobertura.

Bom os nossos testes foram divididos em 4 arquivos sendo eles:

test_esqueleto.py - esse é o primeiro teste que criamos, e ele cria dois competidores e testa de se o registro de um vencedor está sendo feito corretamente ou não, se está iniciando uma luta corretamente, está registrando as tomadas de tempo da categoria seguidor de linha corretamente.

test_competicoes.py - esse vai criar um evento e adicionar robôs a esse evento, verificando se a inscrição está ocorrendo corretamente, e tenta registrar a tomada de tempo do seguidor de linha.

test_entidades.py - esse testa a criação de uma equipe, desde o processo de criar um líder, adicionar os membros, e robôs da equipe.

test_usuarios.py - esse testa toda a lógica das classes dos tipos de usuários, aqui está uma das partes mais importantes do nosso código, pois são os usuários que tem acesso para alterar informações do nosso sistema, cada um com suas devidas permissões, portanto aqui se concentra os principais testes do nosso sistema.