
# Descrição Detalhada do Projeto: Sistema de Controle de Competição Esportiva

### Objetivo

Desenvolver um sistema para controle de uma competição esportiva no formato de pontos corridos, entre 6 a 10 equipes ou atletas (futebol, vôlei, tênis, tênis de mesa, etc). O sistema permitirá o gerenciamento das equipes, sorteio e exibição de partidas, registro de resultados, e exibição de gráficos de performance das equipes. O projeto deve ser feito utilizando Python e SQL, com uma interface de linha de comando e visualização de dados em uma aplicação web local.

## Funcionalidades

### 1. Gerenciamento de Equipes

- **Ver equipes cadastradas**: Exibir uma lista de todas as equipes ou atletas cadastrados no sistema, com suas informações detalhadas.
- **Cadastrar equipe**: Permitir a entrada de novas equipes ou atletas no sistema, solicitando nome, localização (estado ou país) e potencial (valor entre 0 e 10 com 2 casas decimais que indica a probabilidade de vitória).
- **Atualizar equipe**: Atualizar as informações de uma equipe ou atleta existente no sistema.
- **Apagar equipe**: Remover uma equipe ou atleta do sistema.

### 2. Sorteio de Partidas

- **Sortear partidas**: Gerar automaticamente uma lista de partidas entre as equipes cadastradas, garantindo que todas as equipes se enfrentem pelo menos uma vez.

### 3. Visualização das Partidas

- **Visualizar partidas**: Exibir uma lista de todas as partidas sorteadas, incluindo as equipes envolvidas e resultado (se disponível).

### 4. Registro de Resultados

- **Jogar a próxima partida**: Registrar o resultado da próxima partida na lista de partidas sorteadas. O sistema deve solicitar os pontos ou gols de cada equipe e calcular o vencedor com base nos pontos informados.

### 5. Visualização de Performance

- **Ver gráfico de performance por equipe**: Mostrar um gráfico de performance para uma equipe selecionada, exibindo a evolução da pontuação ao longo das partidas.

## Detalhes de Implementação

### 1. Banco de Dados (banco_dados.py)

Crie um banco de dados SQLite que armazenará as informações das equipes e partidas. O banco de dados deve conter duas tabelas principais:

- **Equipes**
  - `id`: Identificador único da equipe (chave primária).
  - `nome`: Nome da equipe.
  - `localizacao`: Localização da equipe (estado ou país).
  - `potencial`: Potencial da equipe (potencial em marcar por partida).

- **Partidas**
  - `id`: Identificador único da partida (chave primária).
  - `equipe_a_id`: Identificador da primeira equipe (chave estrangeira referenciando `equipes`).
  - `equipe_b_id`: Identificador da segunda equipe (chave estrangeira referenciando `equipes`).
  - `vencedor_id`: Identificador da equipe vencedora (chave estrangeira referenciando `equipes`).
  - `gols_equipe_a`: Pontuação da primeira equipe.
  - `gols_equipe_b`: Pontuação da segunda equipe.

Crie classes Python para representar equipes e partidas, e métodos estáticos para interagir com o banco de dados. As classes devem incluir métodos para adicionar, atualizar, deletar e recuperar registros do banco de dados.

**Equipe**

- Métodos:
  - `adicionar_equipe(nome, localizacao, potencial)`: Adiciona uma nova equipe ao banco de dados.
  - `apagar_equipe(equipe_id)`: Remove uma equipe do banco de dados.
  - `atualizar_equipe(equipe_id, nome, localizacao, potencial)`: Atualiza as informações de uma equipe.
  - `obter_equipes()`: Retorna uma lista de todas as equipes cadastradas.

**Partida**

- Métodos:

    - `adicionar_partida(equipe_a_id, equipe_b_id, gols_equipe_a, gols_equipe_b)`: Adiciona uma nova partida ao banco de dados e registra o vencedor.
    - `obter_partidas()`: Retorna uma lista de todas as partidas.

**Funções:**

- `menu_principal()`: Exibe o menu principal e captura a escolha do usuário.
- `gerenciar_equipes()`: Exibe um submenu para gerenciar equipes (ver, adicionar, atualizar, apagar).
- `sortear_partidas()`: Sorteia partidas entre as equipes cadastradas.
- `visualizar_partidas()`: Exibe todas as partidas sorteadas.
- `jogar_proxima_partida()`: Solicita a pontuação das equipes para a próxima partida e registra o resultado.
- `ver_grafico_performance()`: Exibe um gráfico de performance para uma equipe selecionada.

**4. Visualização Web**

Utilize os conhecimentos adquiridos em Web para criar uma aplicação web simples que permita visualizar o ranking das equipes e as partidas. A aplicação deve acessar os dados do banco de dados e renderizar templates HTML para exibir as informações.

## Implementação dos Gráficos

Utilize a biblioteca Matplotlib para gerar gráficos de performance das equipes. O gráfico deve mostrar a evolução da pontuação de uma equipe ao longo das partidas.

## Detalhamento Adicional das Partidas

As partidas serão jogadas automaticamente e os resultados serão baseados no potencial das equipes. A quantidade de gols ou pontos de cada equipe será gerada aleatoriamente em função do potencial da equipe, seguindo a fórmula:

```
def gerar_pontuacao(potencial):
    return round(random.uniform(0, potencial), 2)
```

Ao jogar a próxima partida, o sistema deve calcular a pontuação de cada equipe utilizando a função `gerar_pontuacao(potencial)` e registrar os resultados no banco de dados.

## Instruções para Iniciar o Projeto a Partir do Repositório Fornecido

### 1. Clonar o Repositório Original

Primeiramente, os alunos devem clonar o repositório original para a máquina local. Isso pode ser feito utilizando o comando `git clone`. Siga os passos abaixo:

1. Abrir o terminal (ou prompt de comando):
    - No Windows, você pode usar o Git Bash, que vem com a instalação do Git.
    - No macOS ou Linux, você pode usar o terminal padrão.
2. Executar o comando para clonar o repositório:

```
git clone https://github.com/augustobmo/lab_social_projeto_final_06-24.git
```

### 2. Criar um Novo Repositório no GitHub

Para ter um repositório próprio onde os alunos podem fazer commits e push do progresso, é necessário criar um novo repositório no GitHub.

1. Acessar GitHub:
    - Entre na sua conta do GitHub (ou crie uma se ainda não tiver).
2. Criar um novo repositório:
    - No canto superior direito, clique no ícone de "+" e selecione "New repository".
    - Dê um nome ao seu repositório, por exemplo, `meu_projeto_final_06-24`.
    - Opcionalmente, adicione uma descrição.
    - Escolha se o repositório será público ou privado.
    - Clique em "Create repository".

### 3. Conectar o Repositório Local ao Novo Repositório no GitHub

Agora, os alunos precisam configurar o repositório local clonado para apontar para o novo repositório que criaram no GitHub.

1. Navegar até o diretório do repositório clonado:
    - No terminal, mude o diretório para onde o repositório foi clonado.

2. Copiar código

```
cd lab_social_projeto_final_06-24
```

3. Adicionar o novo repositório como o repositório remoto:
    - Use o comando `git remote add` para adicionar o novo repositório como remoto.

```
git remote rename origin upstream
git remote add origin https://github.com/seu_usuario/meu_projeto_final_06-24.git
```

4. Verificar se a configuração está correta:
    - Execute o comando abaixo para verificar se os repositórios remotos foram configurados corretamente.

```
git remote -v
```

5. Fazer o primeiro push para o novo repositório:
    - Envie os dados para o novo repositório no GitHub.

```
git push -u origin master
```

**4. Iniciar o Desenvolvimento**

Com o repositório configurado, os alunos podem agora começar o desenvolvimento do projeto. Algumas etapas iniciais incluem:

1. Instalar Dependências:
2. Estruturar o Projeto:
    - Familiarizar-se com a estrutura do projeto e os arquivos já existentes.
    - Criar um plano de ação com base na descrição do projeto e as funcionalidades que precisam ser implementadas.
3. Implementar Funcionalidades:
    - Começar a adicionar e testar funcionalidades conforme descrito no enunciado do projeto.
    - Fazer commits frequentes com mensagens claras.
    - Utilizar branches para novas funcionalidades e depois fazer merge para a branch principal (`master` ou `main`).
4. Atualizar o Repositório Remoto:
    - Periodicamente, faça push das mudanças para o repositório no GitHub para manter um backup atualizado e permitir colaboração se estiverem trabalhando em grupo.

## Critérios de Avaliação

| Item | Descrição | Peso (0-1) |
|---|---|---|
| **Funcionalidade de Cadastro de Equipes** | Implementação completa das funcionalidades de ver, adicionar, atualizar e apagar equipes. | |
| **Funcionalidade de Sorteio de Partidas** | Sorteio de partidas de forma aleatória e garantindo que todas as equipes se enfrentem. | |
| **Registro de Resultados** | Implementação da função para registrar resultados das partidas e determinar o vencedor. | |
| **Visualização de Partidas** | Visualização correta de todas as partidas sorteadas com resultados. | |
| **Visualização de Gráficos de Performance** | Gráficos de performance das equipes mostrando a evolução da pontuação. | |
| **Banco de Dados** | Implementação correta do banco de dados com as tabelas e relacionamentos descritos. | |
| **Interação com Banco de Dados** | Métodos para adicionar, atualizar, deletar e recuperar dados funcionando corretamente. | |
| **Interface de Linha de Comando** | Menu principal e submenus funcionais para gerenciamento de equipes e partidas. | |
| **Interface Web** | Aplicação web funcional para visualização de ranking e partidas. | |

| | | |
|---|---|---|
| **Simulação de Partidas** | Algoritmo de simulação de partidas | |
| **Pontos Extras** | | |
| **Sistema de Notificações** | Notificações sobre próximos jogos, resultados e atualizações na página web com auto carregamento da página. | |
| **Relatórios Detalhados** | Geração de relatórios detalhados em PDF e Excel. | |
| **Customização opcional** | Crie alguma funcionalidade diferente das solicitadas. | |