# DOCUMENTAÇÃO MAHINDRA FUSION

*( python )*

---


- **Integrantes:**
  - Caio Felipe de Lima Bezerra *RM: 556197*
  - Gabriel Terra Lilla dos Santos *RM: 554575*
  - Lucas Derenze Simidu *RM: 555931*
  - Marcos Vinícius da Silva Costa *RM: 555490*
  - Ricardo Cerazi Di Tilia *RM: 555155*

# 1. Descrição do Projeto

O sistema é uma plataforma de apostas voltada para corridas de Fórmula E, permitindo que os usuários criem contas, explorem informações sobre NFTs (Tokens Não Fungíveis) relacionados às corridas e façam apostas em eventos.

Os usuários podem visualizar detalhes de diferentes NFTs, acessar informações sobre as corridas e apostar em seus corredores favoritos.

O sistema também inclui funcionalidades de validação, como verificação de idade e confirmação de dados pessoais, além de mostrar resultados de apostas realizadas.

# 2. Objetivo do Projeto

O objetivo do sistema é proporcionar uma experiência interativa e envolvente para os usuários que desejam participar de apostas em corridas de Fórmula E, ao mesmo tempo em que exploram NFTs associados, assim contribuindo para a propagação de sua imagem, fazendo um sistema totalmente interativo e atrativo para o público consumidor;

**Ele busca:**

- ➔ **Facilitar a Criação de Contas:** Permitir que novos usuários criem contas facilmente, garantindo a segurança e a integridade de seus dados;
- ➔ **Oferecer Informações Abrangentes:** Fornecer detalhes sobre os NFTs e as corridas, ajudando os usuários a tomar decisões informadas ao realizar apostas;
- ➔ **Permitir Apostas Seguras e Divertidas:** Garantir que as apostas sejam feitas de maneira fácil, com um sistema que calcula odds e exibe resultados de forma clara;
- ➔ **Promover um Ambiente de Apostas Responsáveis:** Validar a idade dos usuários e fornecer feedback sobre suas apostas, incentivando uma participação consciente e responsável nas corridas;

Interação Visual e Informativa: Utilizar elementos gráficos e interativos para melhorar a experiência do usuário e facilitar a navegação pelo sistema;

# 3. Instalação de pré-requisitos

Para usufruir do sistema com êxito, é necessário possuir conhecimento da versão do Python que foi utilizada para o desenvolvimento, nesse caso, sendo o "***Python 3.12.2***";

Também é necessário a instalação de certas bibliotecas para o funcionamento magistral do sistema, sendo eles:

- ➔ *Pandas*, comando: ***pip install pandas***
- ➔ *Tabulate*, comando: ***pip install tabulate***

# 4. Organização de Arquivos

A organização dos arquivos no repositório do sistema Mahindra Fusion, desenvolvido em Python, segue a seguinte organização de arquivos:

```
MAHINDRA_FUSION_PYTHON/
├── bet.py
├── database.py
├── function.py
├── ilustration.py
├── LICENSE
├── main.py
└── README.md
```

## 4.1. main.py

O arquivo ***main.py*** implementa o sistema principal da aplicação, utilizando de menu principal para a área de compras de NFTs (tokens não fungíveis) ligados a corridas e onde importa as funcionalidades de apostas em corridas fictícias da ***bet.py***, também permitindo a criação e gerenciamento de contas de usuário;

**Menu Principal**: O usuário pode acessar a conta, saber mais sobre o sistema, entrar no mercado de NFTs ou acessar um sistema de apostas;

**Opções:**

- ➔ **Sua conta:** Ver ou criar uma conta de usuário, onde são armazenados dados como nome, idade, CPF, email e telefone;

- ➔ **Sobre Nós:** Apresenta um resumo sobre o serviço;

- ➔ **BET:** Permite ao usuário acessar um sistema de apostas, mas só é acessível para maiores de idade;

- ➔ **Mercado Virtual (Fusion Market):** Oferece NFTs relacionados a corridas, com detalhes sobre cada NFT e a possibilidade de visualizar informações adicionais e gráficos;

- ➔ **Criação e Verificação de Conta:** O sistema solicita e valida os dados do usuário para criar uma conta, como nome, idade, CPF e email;

  Se a conta já existir, os dados são exibidos;
  Se o usuário for menor de idade, ele é redirecionado ao menu;

- ➔ **Mercado de NFTs:** O usuário pode visualizar uma lista de NFTs disponíveis para compra ou visualização.

  Detalhes de cada NFT são mostrados, incluindo informações sobre o piloto, veículo, velocidade e corrida, além de permitir visualizações gráficas;

## 4.2. bet.py

Este arquivo ***bet.py*** implementa um sistema de apostas fictícias para 5 corridas de *Fórmula E* com suas credenciais geradas aleatoriamente a cada execução, utilizando uma interface simples e eficaz de comando onde o usuário pode escolher entre essas corridas, realizar suas apostas a partir da escolha de um corredor e um valor, então visualizar seus resultados. O sistema se destaca pelo cálculo das probabilidades (odds), a randomização dos resultados e a simulação de corridas com diferentes competidores.

**Importações e Dependências:** O arquivo utiliza bibliotecas como:

- ➔ ***random*** para randomizar a posição dos corredores.

- ➔ ***pandas*** para organizar e exibir as corridas em forma de tabela.

- ➔ ***tabulate*** para melhorar a apresentação das tabelas no terminal.

- ➔ Outras funções e dados são importados de arquivos externos (***database, function, ilustration***), fornecem funcionalidades adicionais, como ilustrações e dados sobre os competidores e corridas.

**Apostas:** As apostas são armazenadas nos dicionários apostas (para acompanhar as apostas atuais) e "*apostas_realizadas"* (para registrar as corridas em que o usuário já apostou, evitando duplicidade);

**O processo de apostar envolve:** Exibir uma lista dos competidores da corrida, junto com suas posições e odds (probabilidades de ganho);

Solicitar ao usuário que escolha um corredor (com base em sua posição) e o valor da aposta;

Registrar essa aposta, incluindo o corredor escolhido, o valor e as odds calculadas;

**Cálculo das Odds:** A função "*calcular_odds"* ajusta as odds (probabilidades de ganho) com base na posição do corredor. Corredores com melhores posições têm odds mais baixos, enquanto aqueles com posições inferiores possuem odds maiores, refletindo um risco maior e, portanto, uma recompensa potencial maior;

A fórmula usada para as odds é "*round(10.0 - (posicao - 1) * (8.5 / 19), 2)*", que diminui as odds gradualmente com base na posição do corredor;

**Simulação da Corrida:** A função "*exibir_corrida*" embaralha a lista de corredores para simular a corrida, gerando uma tabela com os competidores, suas posições e odds.
Se o usuário já apostou naquela corrida, ele é notificado que não pode apostar novamente;

Caso contrário, o usuário é solicitado a fazer uma aposta, escolhendo a posição do corredor e o valor em dinheiro;

**Exibição dos Resultados:**A função "*mostrar_resultados*" simula o resultado final de cada corrida de forma aleatória (usando random.randint para determinar a posição final do corredor);

O usuário ganha a aposta se o corredor escolhido terminar em uma das três primeiras posições;

O lucro é calculado multiplicando o valor apostado pelas odds do corredor. Se o usuário perder, ele recebe uma mensagem correspondente;

**Corridas Disponíveis:** As corridas são pré-definidas no programa e incluem eventos como:

1. São Paulo E-Prix
2. Mônaco E-Prix
3. Nova Iorque E-Prix
4. Berlim E-Prix
5. Londres E-Prix

Cada corrida tem sua própria função, e ao ser selecionada pelo usuário, chama a função "*exibir_corrida*" para iniciar o processo de apostas;

**Navegação no Menu:** O menu principal ("bet_menu") permite ao usuário selecionar uma corrida para apostar ou visualizar os resultados das apostas feitas;

Há também a opção de retornar ao menu principal do programa (importado do módulo main), o que indica que esse sistema de apostas pode ser parte de um programa maior;

A interação com o menu é feita via input do usuário, e a navegação continua até que o usuário escolha sair ou retornar;

**Fluxo de Retorno:** Após cada ação, como apostar ou visualizar os resultados, o usuário é levado de volta ao menu de apostas através da função "voltar_bet", que pausa o programa até o usuário pressionar ENTER, limpando a tela e exibindo o menu novamente;

**Funções principais:**

- ➔ **calcular_odds("*posicao*"):** Calcular as odds com base na posição do corredor;

- ➔ **“exibir_corrida(corrida, nome_corrida)”:** Exibe a corrida com os competidores, suas posições e odds, e permite que o usuário faça apostas;

- ➔ **mostrar_resultados(nft):** Exibe o resultado final das corridas e calcula o ganho ou perda do usuário com base nas apostas feitas;

- ➔ **corrida1, corrida2, corrida3, corrida4, corrida5(nft):** Funções específicas para cada corrida, que chamam exibir_corrida e permitem a navegação no menu;

- ➔ **bet_menu(nft):** Menu principal onde o usuário escolhe a corrida ou visualiza os resultados;

## 4.3. database.py

Este arquivo database.py parece ser parte de um projeto voltado para interagir com **NFTs** relacionados à *Fórmula E*, além de simular apostas em corridas.

**Índices e opções:** Definição de listas que contêm os índices dos menus e as opções de apostas disponíveis;

**Produtos (NFTs):** Um dicionário chamado produtos contém informações sobre NFTs relacionados a corridas de Fórmula E.

Cada NFT possui atributos como:

- ➔ ID, nome, quantidade, equipe, veículo, piloto, velocidade, corrida, pista, volta e preço;
- ➔ Há também um atributo de desenho gráfico (desenho) associado a cada NFT, que exibe representações visuais coloridas;

**Corridas:** Um dicionário chamado corridas lista diferentes eventos de corrida, contendo equipes e nomes de pilotos para simular os participantes.

**Texto e explicações:** Vários textos explicativos são incluídos, como o título gráfico do projeto, uma introdução explicando a finalidade do projeto Mahindra Fusion (em parceria com a Tech Mahindra), o problema de baixa popularidade da Fórmula E, a solução proposta para criar um site interativo e gamificado, e os objetivos esperados com o projeto, como aumentar a visibilidade e engajamento da audiência.

**Funções relacionadas a NFTs:** Funções como "*imprimir_descricao_nft*", "*nft_mais_detalhes*", "*nft_grap*", e "*nft_voltar*" permitem que o usuário visualize as informações dos NFTs, detalhes adicionais (como código e preços), representações gráficas e retorne ao menu anterior.

**Menu de Produtos:** Há uma função ("*produtos_market*") que exibe os produtos disponíveis no mercado virtual de NFTs.

## 4.4. function.py

O arquivo ***function.py*** contém funções auxiliares para uma aplicação em Python, provavelmente uma interface de linha de comando (CLI), que realiza operações como solicitar e validar dados do usuário, exibir informações, limpar a tela e encerrar o programa.
Aqui está um resumo das principais funções:

- ➔ **solicitar_dado(mensagem, funcao_validacao):** Solicita um dado do usuário com uma mensagem específica e utiliza uma função de validação fornecida como argumento para garantir que o dado está no formato correto;

- ➔ **imprime_dados_conta(dados):** Exibe de forma formatada os dados de uma conta (presumivelmente um dicionário), apresentando cada chave e valor de forma clara;

- ➔ **mostrar_resumo():** Mostra um resumo de informações pré-definidas (como introdução, problema, solução, objetivos, justificativa e conclusão). Essas variáveis não estão declaradas no código apresentado, mas supõe-se que sejam globais ou importadas de outro módulo;

- ➔ **escolha(opcoes, msg):** Solicita uma escolha do usuário entre várias opções disponíveis e repete a solicitação até que o usuário insira uma opção válida;

- ➔ **verifica_numero(msg):** Solicita um número ao usuário é válida se a entrada é numérica, repetindo a solicitação até que seja fornecido um número válido;

- ➔ **nao_vazio(msg):** Válida que o usuário insira um valor não vazio;

- ➔ **limpar_tela():** Limpa a tela do terminal, utilizando o comando apropriado dependendo do sistema operacional (Windows ou outros);

- ➔ **sair_programa():** Limpa a tela e exibe uma mensagem de finalização do programa;

Essas funções são projetadas para facilitar a interação com o usuário e garantir que os dados inseridos sejam válidos e que a interface seja mantida limpa durante a execução.

## 4.5. ilustration.py

O arquivo **illustration.py** é um script que define representações artísticas em ASCII art, aplicando cores diferentes para cada uma das artes criadas;

Ele inclui desenhos que parecem representar logotipos ou símbolos de marcas de corrida de Fórmula E e funções que imprimem essas artes com cores específicas no terminal;

Artes em ASCII: Existem várias representações de carros ou logotipos de equipes de Fórmula E, como:

- ➔ DS Techeetah (art1);

- ➔ Audi Schaeffler (art2);

- ➔ Jaguar TCS Racing (art3);

**Cores personalizadas:** Para cada arte, há a aplicação de uma cor específica no terminal:

- ➔ art1 é colorida em amarelo;

- ➔ art2 é colorida em vermelho;

- ➔ art3 é colorida em azul-turquesa;

**Funções de logo com arte em ASCII:**

- ➔ mahindra_fusion_logo(): Desenha e imprime o logotipo da Mahindra, aplicando a cor verde-limão;

- ➔ fusion_market_logo(): Imprime um logotipo com o texto "Fusion Market", também em verde-limão;

- ➔ fusion_bet_logo(): Imprime o logotipo da "Fusion Bet", novamente em verde-limão;

As funções são responsáveis por desenhar e exibir logotipos específicos com formatação colorida, sendo úteis em contextos de apresentação de equipes ou marcas em ambientes de console.