## Trabalho Final – Desenvolvimento de um Bot de Investimentos com a API da Binance

O desafio será desenvolver um **Bot de Investimentos Simulado** (*paper trading*), que se conectará à plataforma de criptomoedas Binance para tomar decisões de compra e venda com base em uma estratégia pré-definida.

## Objetivo

Desenvolver um aplicativo em Python (interface via console ou gráfica) que simule operações de trade de criptomoedas. O bot deverá obter dados de mercado em tempo real através da API oficial da Binance, aplicar uma estratégia de investimento baseada em médias móveis e registrar todas as operações realizadas em um arquivo local.

**Importante:** O bot operará em modo de simulação. **Não utilizaremos dinheiro real.** O objetivo é puramente acadêmico, focando na arquitetura do software.

## Requisitos Funcionais

1. **Interface de Configuração e Interação:**
    - O usuário deverá poder inserir suas chaves de API da Binance (é **obrigatório** o uso da **Binance Testnet**, a rede de testes, para não envolver valores reais).
    - Deverá haver campos para o usuário definir:
        - O par de moedas a ser negociado (ex: `[BTC, USDT],[ETH,USDT]`).
        - O "capital" inicial para a simulação (ex: 1000 USDT).
        - Os parâmetros da estratégia (ex: os períodos das duas médias móveis).
    - Comandos (ou botões, caso opte por GUI) para "Iniciar Bot" e "Parar Bot".
2. **Conexão e Coleta de Dados:**
    - O sistema deve se conectar à API da Binance para buscar o histórico de preços do par de moedas selecionado.
    - Enquanto o bot estiver ativo, ele deve continuar buscando o preço atual em intervalos regulares (ex: a cada 1 minuto).
3. **Lógica da Estratégia de Trading:**
    - A estratégia base será o **cruzamento de médias móveis (Moving Average Crossover)**.
    - **Sinal de Compra:** Quando a média móvel curta (ex: 10 períodos) cruzar para *cima* da média móvel longa (ex: 30 períodos).
    - **Sinal de Venda:** Quando a média móvel curta cruzar para *baixo* da média móvel longa.
    - O bot só pode comprar se tiver capital (USDT) e só pode vender se tiver o criptoativo.
    - Para referência, leia sobre a estratégia em: Investopedia: Moving Average Crossover
4. **Simulação de Operações (Paper Trading):**

    - O sistema deve gerenciar um portfólio virtual, controlando a quantidade de capital (USDT) e de criptoativo (ex: BTC).
    - Ao receber um sinal de compra, o bot deve simular a compra do máximo possível do criptoativo com o capital disponível.
    - Ao receber um sinal de venda, o bot deve simular a venda de toda a sua posição no criptoativo.
5. **Logging e Exibição de Resultados:**
    - Todas as operações (compra e venda) devem ser registradas em um arquivo (`trades.json` ou `trades.csv`), contendo: data/hora, tipo (COMPRA/VENDA), preço e quantidade.
    - A interface (console ou gráfica) deve exibir em tempo real:
        - O saldo atual do portfólio (ex: `500.50 USDT` e `0.015 BTC`).
        - O valor total do portfólio em USDT (capital + valor do criptoativo ao preço atual).
        - Um log ou uma tabela com as últimas operações realizadas.

## Requisitos Não Funcionais e Técnicos

1. **Linguagem:** Python .
2. **Orientação a Objetos (Essencial):** O código deve ser orientado a objetos.Com clara separação de responsabilidades em classes, como por exemplo:
    - Uma classe para gerenciar a comunicação com a API da Binance (`BinanceClient`).
    - Uma classe base abstrata para a estratégia (`TradingStrategy`) e uma classe filha que a implementa (`MovingAverageStrategy`).
    - Uma classe para gerenciar o portfólio simulado (`Portfolio`).
    - Uma classe principal para a aplicação e a interface (`App`).
3. **Interface:** A interação com o usuário pode ser via terminal (console). Opcionalmente, pode ser desenvolvida uma interface gráfica (sugestão: Tkinter).
4. **Persistência de Dados:** As operações de trade devem ser salvas em um arquivo local (JSON ou CSV).

## Sugestões Técnicas

- **API da Binance:** Recomendo o uso da biblioteca `python-binance`, que simplifica a comunicação com a API.
- **Binance Testnet:** Isso fornecerá chaves de API para um ambiente de teste que funciona exatamente como o real, mas com dinheiro fictício.
- **Cálculo das Médias Móveis:** Bibliotecas como `pandas` ou `numpy` são eficientes para calcular indicadores técnicos a partir de séries de dados de preços.

## Entregáveis

1. **Código-fonte completo** do projeto, organizado em módulos.
2. Diagrama de Classe UML

3. Arquivo `README.md` explicando o projeto, a estrutura das classes e como executar a aplicação.

## Critérios de Avaliação (Total: 10,0 pontos)

- **Modelagem e Estrutura OO (4,0 pontos):** Correta aplicação de classes, herança/composição, encapsulamento e separação de responsabilidades.
- **Diagrama de Classes UML (1,5 pontos):** Clareza e representação fiel do sistema.
- **Integração com a API Binance (1,5 pontos):** Conexão, busca de dados e tratamento de erros.
- **Implementação da Estratégia de Trading (1,0 pontos):** Lógica de cálculo dos indicadores e geração de sinais de compra/venda.
- **Interface com o Usuário (Console ou Gráfica) (1,0 pontos):** Usabilidade, clareza na exibição das informações e funcionalidade dos controles/comandos.
- **Persistência e Logging de Dados (1,0 ponto):** Armazenamento correto e consistente das operações.