

UNIVERSIDADE FEDERAL DA INTEGRAÇÃO LATINO-AMERICANA – UNILA
CURSO DE ENGENHARIA FÍSICA

YAN COSTA CARNEOSSO

AUTOMAÇÃO FINANCEIRA EM FINANÇAS DESCENTRALIZADAS (DEFI): ANÁLISE QUANTITATIVA DE SÉRIES FINANCEIRAS E SISTEMA DE AUTOMAÇÃO FINANCEIRA EM OPERAÇÕES COM CRIPTOMOEDAS.

FOZ DO IGUAÇU - PR

2025

Versão Final Homologada
27/12/2025 13:45

YAN COSTA CARNEOSSO

AUTOMAÇÃO FINANCEIRA EM FINANÇAS DESCENTRALIZADAS (DEFI): ANÁLISE QUANTITATIVA DE SÉRIES FINANCEIRAS E SISTEMA DE AUTOMAÇÃO FINANCEIRA EM OPERAÇÕES COM CRIPTOMOEDAS.

Trabalho de Conclusão do Curso, como requisito obrigatório para a obtenção do título de bacharel em Engenharia Física pela Universidade Federal da Integração Latino-Americana (UNILA).

Orientador: Profº Drº Johan Alexander Cortés Suárez

Universidade Federal da Integração Latino-Americana

Curso de Engenharia Física

Orientador: Profº Drº Johan Alexander Cortés Suárez

Foz do Iguaçu - PR

2025

Versão Final Homologada
27/12/2025 13:45

Carneosso, Y. C.
Automação financeira em Finanças Descentralizadas (DeFi):
Análise quantitativa de séries financeiras e sistema de automação financeira em operações com criptomoedas./ Yan Costa Carneosso. – Foz do Iguaçu - PR, 2025-

138p. : il. (algumas color.) ; 30 cm.

Orientador: Profº Drº Johan Alexander Cortés Suárez
Trabalho de Conclusão de Curso – Universidade Federal da Integração Latino-Americana
Curso de Engenharia Física
, 2025.

1. Finanças Descentralizadas. 2. Automação Financeira. 3. Contratos Inteligentes.
I. Orientador Drº Johan Alexander Cortés Suárez. II. Universidade Federal da Integração Latino-Americana. III. Faculdade de Engenharia Física IV. Automação financeira em Finanças Descentralizadas (DeFi): Análise quantitativa de séries financeiras e sistema de automação de contratos inteligentes.

Versão Final Homologada
27/12/2025 13:45

YAN COSTA CARNEOSSO

AUTOMAÇÃO FINANCEIRA EM FINANÇAS DESCENTRALIZADAS (DEFI): ANÁLISE QUANTITATIVA DE SÉRIES FINANCEIRAS E SISTEMA DE AUTOMAÇÃO FINANCEIRA EM OPERAÇÕES COM CRIPTOMOEDAS.

Trabalho de Conclusão do Curso, como requisito obrigatório para a obtenção do título de bacharel em Engenharia Física pela Universidade Federal da Integração Latino-Americana (UNILA).

Orientador: Profº Drº Johan Alexander Cortés Suárez

Trabalho aprovado. Foz do Iguaçu - PR, 12 de dezembro de 2025

Documento assinado digitalmente
gov.br JOHAN ALEXANDER CORTES SUAREZ
Data: 22/12/2025 23:27:27-0300
Verifique em https://validar.iti.gov.br

Profº Drº Johan Alexander Cortés Suárez
Orientador

Documento assinado digitalmente
gov.br ALEXANDER ARGUELLO QUIROGA
Data: 18/12/2025 18:28:06-0300
Verifique em https://validar.iti.gov.br

Professor Alexander Arguello Quiroga
Convidado

Documento assinado digitalmente
gov.br ALTEMIR BORTULI JUNIOR
Data: 19/12/2025 09:15:59-0300
Verifique em https://validar.iti.gov.br

Professor Altemir Bortuli Junior
Convidado

Foz do Iguaçu - PR
2025

Versão Final Homologada
27/12/2025 13:45

Agradecimentos

Agradeço primeiramente ao meu pai e à minha mãe, Sandei Luis Carneosso e Fernanda Régia da Costa, que me mostraram o caminho, me apresentaram o mundo por meio do conhecimento e sempre agiram como exemplos de luta, honra e lealdade. Agradeço pela minha criação como filho e pela formação como cidadão. Ofereço-lhes, pai e mãe, o reconhecimento do meu trabalho e de todas as conquistas que alcancei e alcançarei, pois, mais que mérito próprio, sou fruto de uma vida de esforços, dedicação e muito amor.

À minha namorada e companheira Beatriz Soares de Paula: sem sua ajuda, incentivo e dedicação, a conclusão deste trabalho, que representa o encerramento de um ciclo, não seria possível. Em diversos momentos durante esta graduação, você foi a minha luz e a minha bússola; obrigado por me mostrar o caminho. Dedico a você todos os possíveis agradecimentos. Aproveito a oportunidade, neste trabalho que será deixado à posteridade, para declarar minha ardente paixão por você, Beatriz. Que essa chama possa queimar ao longo de muitos anos e que o calor das nossas lembranças esteja sempre conosco.

Aos meus professores da universidade, em especial àqueles que me orientaram em projetos de pesquisa e extensão, sou grato pela dedicação, pelos ensinamentos e pela confiança depositada no meu trabalho. Agradeço também aos professores que fizeram parte do meu ensino básico, que contribuíram de forma essencial para minha formação e despertaram em mim a vontade de aprender.

Aos meus amigos, só tenho a agradecer pelos bons momentos que dividimos ao longo dos últimos anos. Foi uma honra existir nesse mesmo "tempo e espaço" que vocês; sem dúvidas, muitas das aventuras que vivemos serão memórias inesquecíveis que terei o prazer de relembrar nos próximos anos. Em especial, agradeço aos meus amigos Kaique de Sotti Silva, por todo o apoio acadêmico e por todas as oportunidades divididas comigo; Marielle Bastos, pelos longos anos de companhia e pelo exemplo de empoderamento e senso de justiça; Denise Rigolon Lorentti e, ao meu amigo e irmão, Marcus Vinicius Wenceslau de Souza Floriano, pelos longos anos de amizade tivemos e por vocês serem parte da minha família em Foz do Iguaçu.

Não poderia deixar de agradecer também ao meu grupo de amigos Arthur Padovan, Bruno Silva da Cruz e Adriana Arami: obrigado pelas inenarráveis aventuras que vivemos ao longo dos últimos anos. Com certeza, os capítulos da minha história na Engenharia Física não seriam tão inesperadamente marcantes e surreais sem vocês.

Por fim, agradeço ao acaso, à aleatoriedade e à sorte. Sem esses fatores, eu jamais ingressaria em um curso como a Engenharia Física, jamais viveria todas as alegrias que vivi. Sem desconsiderar todo o esforço necessário, é graças ao acaso que tenho o orgulho de apresentar este trabalho; é graças à aleatoriedade que me tornarei engenheiro físico e, se a sorte me permitir, seguirei contribuindo para o avanço da universidade pública, gratuita e de qualidade para todos.

Versão Final Homologada
27/12/2025 13:45

AVISO

Este trabalho foi desenvolvido exclusivamente com fins acadêmicos, tendo como objetivo a análise conceitual e técnica dos temas abordados. As metodologias, modelos, algoritmos e estratégias apresentados não constituem recomendação de investimento ou orientação financeira, tampouco devem ser aplicados diretamente em mercados reais sem as devidas adaptações, validações adicionais e consideração de riscos operacionais, financeiros e regulatórios.

Qualquer aplicação prática dos conteúdos aqui discutidos é de inteira responsabilidade do leitor.

Versão Final Homologada
27/12/2025 13:45

Resumo

Este estudo propõe uma revisão bibliográfica sobre o conceito de Finanças Descentralizadas *(DeFi)*, seu surgimento, funcionamento, os contrastes entre esse tipo de operação e aquelas realizadas por bancos e outras instituições financeiras tradicionais, bem como as possíveis oportunidades e motivações existentes nesse mercado. Foram explorados instrumentos de automatização financeira aplicados a esse cenário, utilizando programação em *Python* para a gestão e o controle automatizado dos sistemas. Modelos matemáticos e estratégias utilizadas no mercado financeiro tradicional foram aplicados para a obtenção de dados e análise quantitativa. A partir dos testes realizados, considerou-se o comportamento histórico do mercado, a fim de simular a aplicação dos métodos propostos em intervalos arbitrários (*backtest*). A finalidade deste trabalho é apresentar oportunidades de desenvolvimento em *DeFi*, validar o uso de estratégias quantitativas como alternativa ao *holding* e demonstrar três mecanismos práticos relacionados à automatização financeira, sendo eles: (i) a construção de um *software* para automação de operações financeiras baseado em gatilhos de operação (*SwapBot*); (ii) um livro de ordens para agendamento de ordens de compra e venda por meio de corretoras descentralizadas (DEX); e (iii) um sistema de *backtest*. A arquitetura dos sistemas foi construída de forma modular, a fim de permitir ao usuário realizar atualizações e aplicar novos modelos por meio de alterações pontuais, mantendo todo o restante inalterado. Também foi considerada a padronização da expressão lógica recebida pelo *backtest* e pelo *SwapBot*, permitindo o intercâmbio entre esses dois sistemas. Dessa forma, uma vez que uma estratégia seja validada pelo *backtest*, ela pode ser imediatamente importada para o *SwapBot*. Os dados históricos foram obtidos por meio da conexão via API entre o *software* desenvolvido e a base de dados disponibilizada pelo *yFinance*. Para a execução das estratégias e a simulação das operações síncronas realizadas, foi utilizada a rede de testes da Binance, a *Binance Testnet*. Com o objetivo de demonstrar a aplicação da metodologia proposta, a estratégia de cruzamento de médias móveis foi testada e analisada em três variações: o cruzamento de médias móveis simples, o ajuste exponencial e a aproximação linear via mínimos quadrados. Os resultados obtidos evidenciam que a integração de sistemas automatizados guiados por estratégias quantitativas pode apresentar melhores resultados do que operações realizadas de forma não automatizada. Para todos os *softwares* desenvolvidos, foram criadas interfaces gráficas, a fim de permitir a utilização do sistema por um maior número de usuários.

Palavras-chave: Finanças Descentralizadas; Automação Financeira; Contratos Inteligentes..

Versão Final Homologada
27/12/2025 13:45

Abstract

Abstract

This study presents a literature review on the concept of Decentralized Finance *(DeFi)*, its emergence, operation, and the contrasts between this type of activity and those conducted by banks and other traditional financial institutions, as well as the potential opportunities and motivations that exist within this market. Financial automation tools applicable to this context were examined, employing *Python* programming for system management and automated control. Mathematical models and strategies commonly used in the traditional financial market were applied for data acquisition and quantitative analysis. Based on the tests performed, the historical behavior of the market was considered in order to simulate the application of the proposed methods over arbitrary intervals (*backtesting*). The purpose of this work is to present development opportunities within the *DeFi* ecosystem, validate the use of quantitative strategies as an alternative to asset holding, and demonstrate three practical mechanisms related to financial automation: (i) the development of a *software* tool for financial operation automation based on execution triggers (*SwapBot*); (ii) an order book designed for scheduling buy and sell orders through decentralized exchanges (DEX); and (iii) a *backtesting* system. The system architecture was designed in a modular manner, allowing users to implement updates and apply new models through targeted modifications while keeping the remaining components unchanged. The standardization of the logical expressions processed by both the *backtest* module and the *SwapBot* was also considered, enabling interoperability between the two systems. Thus, once a strategy is validated through the *backtesting* module, it can be immediately imported into the *SwapBot*. Historical data were obtained through an API connection between the developed *software* and the dataset provided by *yFinance*. To execute the strategies and simulate synchronous operations, the Binance testing environment, the *Binance Testnet*, was employed. To illustrate the application of the proposed methodology, the moving average crossover strategy was tested and evaluated in three variations: simple moving average crossover, exponential adjustment, and linear approximation via least squares. The results indicate that the integration of automated systems guided by quantitative strategies can yield superior performance compared to non-automated operations. Graphical interfaces were developed for all *software* tools created, ensuring accessibility and facilitating system use by a broader range of users.

Keywords: Decentralized Finance; Financial Automation; Smart Contracts.

Versão Final Homologada
27/12/2025 13:45

Lista de ilustrações

Figura 1 – Ilustração dos pontos de cruzamento entre a média móvel rápida e a média móvel lenta em um gráfico de preços. 27

Figura 2 – Evolução histórica do valor total bloqueado (TVL) no ecossistema *DeFi* entre 2018 e 2025. 39

Figura 3 – Evolução do valor de mercado (market cap) das stablecoins entre 2018 e 2025. 43

Figura 4 – Evolução do valor total tokenizado em ativos do mundo real (RWA) por categoria, entre 2018 e 2025. 44

Figura 5 – Distribuição percentual do valor total alocado em diferentes categorias de ativos tokenizados. 45

Figura 6 – Evolução temporal do valor total (market cap) de ativos do mercado RWA, segmentados por categoria. 46

Figura 7 – Estrutura lógica utilizada para identificar eventos de cruzamento entre médias móveis rápida e lenta. 53

Figura 8 – Estrutura lógica utilizada para identificar oportunidades de operações agendadas no Livro de Ordens. 54

Figura 9 – Estrutura de arquivos gerada após a execução dos testes de *backtest*. 55

Figura 10 – Arquitetura operacional do sistema de backtest. 56

Figura 11 – Arquitetura operacional do sistema de trading automatizado. 59

Figura 12 – Arquitetura operacional do módulo de Livro de Ordens. 60

Figura 13 – Primeira versão da interface de operação. 63

Figura 14 – Interface final do sistema. 64

Figura 15 – Interface do módulo de backtest. 65

Figura 16 – Página inicial da dashboard. 67

Figura 17 – Página da EMA na *dashboard*. 68

Figura 18 – Comparação temporal das estratégias baseadas em médias móveis de 2018 a 2022. 69

Figura 19 – Comparação temporal das estratégias baseadas em médias móveis de 2021 a 2022. 70

Versão Final Homologada
27/12/2025 13:45

Lista de tabelas

Tabela 1 – Desempenho dos indicadores técnicos nos diferentes períodos analisados. 71
Tabela 2 – Diferenças qualitativas entre as médias móveis 73

Versão Final Homologada
27/12/2025 13:45

Lista de abreviaturas e siglas

API	Application Programming Interface — Conjunto de regras que permite a comunicação entre diferentes sistemas e aplicações de forma padronizada.
BCH	Bitcoin Cash — Criptomoeda derivada do Bitcoin, criada com foco em maior capacidade de processamento de transações.
BCG	Boston Consulting Group — Empresa global de consultoria estratégica que atua em negócios, tecnologia e políticas públicas.
BSC	Binance Smart Chain — Blockchain compatível com a EVM, desenvolvida pela Binance para execução de contratos inteligentes com baixo custo.
BTC	Bitcoin — Primeira criptomoeda descentralizada, baseada em blockchain e no mecanismo de consenso *Proof of Work*.
CeFi	Centralized Finance — Modelo financeiro baseado em intermediários centralizados, como corretoras e instituições tradicionais.
CPU	Central Processing Unit — Unidade central de processamento responsável pela execução de instruções e operações computacionais.
CSV	Comma-Separated Values — Formato de arquivo para armazenamento de dados tabulares, utilizando vírgulas como separadores.
DeFi	Decentralized Finance — Ecossistema financeiro descentralizado que utiliza contratos inteligentes em blockchains públicas.
DEX	Decentralized Exchange — Plataforma de negociação descentralizada que permite trocas diretas entre usuários sem intermediários.
EMA	Exponential Moving Average — Média móvel que atribui maior peso aos dados mais recentes, sendo sensível a variações de preço.
ETH	Ethereum — Plataforma blockchain que possibilita a execução de contratos inteligentes e aplicações descentralizadas.
EVM	Ethereum Virtual Machine — Ambiente de execução que permite o funcionamento padronizado de contratos inteligentes no Ethereum.
FED	Federal Reserve System — Banco central dos Estados Unidos, responsável pela política monetária e estabilidade financeira.
GENIUS	Guiding and Establishing National Innovation for U.S. Stablecoins — Proposta regulatória dos EUA voltada à emissão e supervisão de stablecoins.
GPU	Graphics Processing Unit — Unidade de processamento gráfico utilizada para cálculos paralelos, comum em mineração e computação intensiva.

Versão Final Homologada
27/12/2025 13:45

HFT	High-Frequency Trading — Estratégia de negociação automatizada que realiza grande volume de operações em alta velocidade.
HTML	HyperText Markup Language — Linguagem de marcação utilizada na estruturação de páginas e aplicações web.
JSON	JavaScript Object Notation — Formato leve de troca de dados, amplamente utilizado em APIs e sistemas distribuídos.
LSMA	Least Squares Moving Average — Média móvel baseada no método dos mínimos quadrados para suavização de séries temporais.
MA	Moving Average — Indicador técnico utilizado para analisar tendências por meio da média dos preços ao longo do tempo.
PoS	Proof of Stake — Mecanismo de consenso que seleciona validadores com base na quantidade de ativos em posse.
PoW	Proof of Work — Mecanismo de consenso baseado na resolução de problemas computacionais para validação de blocos.
RWA	Real-World Assets — Ativos do mundo real tokenizados e representados em blockchain, como imóveis e títulos financeiros.
SMA	Simple Moving Average — Média móvel aritmética calculada a partir dos preços em um período fixo.
TradFi	Traditional Finance — Sistema financeiro tradicional baseado em instituições centralizadas e regulamentadas.
TVL	Total Value Locked — Métrica que representa o valor total de ativos bloqueados em protocolos DeFi.
USD	United States Dollar — Moeda oficial dos Estados Unidos, amplamente utilizada como referência global.
USDC	USD Coin — Stablecoin lastreada em dólar norte-americano, emitida por entidades reguladas.
USDT	Tether — Stablecoin pareada ao dólar, amplamente utilizada para liquidez no mercado de criptoativos.
VASPs	Virtual Asset Service Providers — Provedores de serviços relacionados a ativos virtuais, como corretoras e custodiante digitais.

Versão Final Homologada
27/12/2025 13:45

Sumário

1	**INTRODUÇÃO**	**17**
2	**FUNDAMENTAÇÃO TEÓRICA**	**23**
2.1	**Automação financeira**	**23**
2.2	**Estratégias Quantitativas**	**26**
2.3	**A origem das criptomoedas**	**27**
2.4	**Ascenção Bitcoin e Blockchain**	**30**
2.4.1	O Surgimento do Bitcoin e a Crise Financeira de 2008.	30
2.4.2	Características Fundamentais do Bitcoin	31
2.4.2.1	Escassez Programada	31
2.4.2.2	Mineração, Incentivos e Dificuldade	32
2.4.2.3	Descentralização e Estrutura de Nós	33
2.4.2.4	Parâmetros de Consenso	33
2.4.2.5	Governança e Atualizações do Protocolo	33
2.4.2.6	Blockchaincomo Infraestrutura	35
2.5	**Contratos inteligentes**	**35**
2.6	**Finanças descentralizadas**	**37**
3	**METODOLOGIA**	**47**
3.1	**Cálculo de médias móveis**	**48**
3.1.1	Médias móveis simples (SMA)	48
3.1.2	Médias móveis com ajuste exponencial (EMA)	48
3.1.3	Médias móveis de mínimos quadrados com aproximação linear (LSMA)	49
3.2	**Arquitetura, Lógica e Mecanismo de Tomada de Decisão do Sistema**	**50**
3.2.1	Arquitetura e funcionamento	50
3.2.2	Lógica Operacional e Processo Decisório	52
3.2.2.1	Metodologia de Operação do Algoritmo	52
3.2.2.2	Etapa 1 – Coleta dos Dados de Mercado	52
3.2.2.3	Etapa 2 – Atualização dos Dados e Indicadores	52
3.2.2.4	Etapa 3 – Aplicação da Estratégia e Tomada de Decisão	53
3.3	**Backtest**	**54**
3.3.1	Descrição	54
3.3.2	Aspectos Técnicos do Desenvolvimento	55

Versão Final Homologada
27/12/2025 13:45

3.4 SwapBot 57
3.4.1 Descrição 57
3.4.2 Aspectos Técnicos do Desenvolvimento 57
3.5 Livro de Ordens 59
3.5.1 Descrição 59
3.5.2 Aspectos Técnicos do Desenvolvimento 60

4 RESULTADOS E DISCUSSÃO 63
4.1 Interface e Experiência de Usuário 63
4.1.1 Interface de Operação 63
4.1.2 Interface do Backtest 65
4.1.3 Dashboard para comparação de performance entre médias móveis 66
4.2 Análise Quantitativa 70
4.3 Análise Qualitativa 72
4.4 Limitações e perspectivas futuras 73

5 CONCLUSÃO 75

REFERÊNCIAS 77

ANEXOS 83

ANEXO A – CÓDIGO-FONTE DO BOT DE TRADING 85

ANEXO B – CÓDIGO DO FONTE BACKTEST 123

Versão Final Homologada
27/12/2025 13:45

1 Introdução

Com o desenvolvimento da internet, muitas atividades humanas já estabelecidas encontraram novas formas de emergir na sociedade. A comunicação, o consumo e os relacionamentos humanos apresentam constantemente novos aspectos propiciados pelas tecnologias emergentes, desde o fim de barreiras geográficas para o consumo virtual até a possibilidade de disseminação de novos ideais de forma global. Com o dinheiro não seria diferente.

Durante as últimas três décadas, houveram diversas tentativas de criar um dinheiro virtual, algo que ultrapassasse a taxonomia dos meios de pagamento digitais e que possuísse as características fundamentais de moeda. Para desempenhar tal papel, a teoria da Economia Monetária, ramo da economia que estuda o comportamento da moeda, seu papel nas transações econômicas, seus efeitos sobre variáveis macroeconômicas e os mecanismos de política monetária, aponta a necessidade de que o ativo em questão exerça as funções de unidade de conta, meio de troca e reserva de valor, além de apresentar propriedades práticas que viabilizem sua durabilidade, portabilidade, divisibilidade, aceitabilidade e verificabilidade (MISHKIN, 2019; MANKIW, 2021).

O projeto mais bem-sucedido nessa missão de desenvolver uma forma de dinheiro virtual é a criptomoeda Bitcoin. Esse criptoativo tem ganhado notoriedade nos últimos anos, pois, apesar de sua elevada volatilidade de preços, suas características computacionais e sua filosofia baseada na descentralização têm se tornado atrativas para o mercado e objeto de crescente atenção acadêmica e institucional (SHARMA, 2019; ANTONOPOULOS, 2015).

Um dos diferenciais para o sucesso do *Bitcoin* (BTC) em relação aos seus antecessores, como *DigiCash*, *B-money* e o *BitGold*, foi o uso da *Blockchain* para o processamento das transações. Nas seções posteriores, será realizada uma revisão bibliográfica sobre a história e os precedentes do BTC, apresentando os aspectos fundamentais de cada projeto utilizado como referência para essa criptomoeda.

Apesar de o BTC ser o maior criptoativo, com capitalização de mercado de aproximadamente 2,3 trilhões de dólares, existem diversos outros criptoativos com características, funcionalidades e aplicações distintas. Em segunda posição, temos o criptoativo *Ether* (ETH), que possui capitalização de mercado de cerca de 542 bilhões de dólares. Sua função é bastante diferente da do BTC: enquanto o *Bitcoin* se propõe a apresentar características de moeda, o ETH tem como premissa atuar como um *token* para custear operações na rede

Versão Final Homologada
27/12/2025 13:45

Ethereum, que funciona como uma máquina virtual para criação de aplicações, contratos inteligentes e processamento de registros.

Com as possibilidades de criação de novos mecanismos financeiros nessas redes, as operações financeiras viabilizadas pela *Blockchain* foram nomeadas como Finanças Descentralizadas (DeFi), cuja principal diferença em relação ao modelo tradicional é a dispensabilidade de um eixo central responsável por gerir, armazenar e honrar os contratos.

O que permite a criação das Finanças Descentralizadas é o uso de contratos inteligentes, conceito criado por Nick Szabo (1996), que o definiu como um protocolo de computador capaz de facilitar, verificar ou executar automaticamente os termos de um contrato. Essa ideia se torna particularmente relevante quando aplicada à rede *Ethereum*, onde os contratos inteligentes funcionam como blocos de código executados de forma descentralizada, permitindo a criação de aplicativos autônomos e eliminando intermediários. O uso de contratos inteligentes também possibilita a interação automatizada na rede. Estimulando um cenário fértil para o desenvolvimento de sistemas de gestão de ativos e para implementação de modelos viáveis de operações de compra e venda no mercado.

No Brasil, embora ainda não exista uma regulamentação específica voltada exclusivamente para *DeFi*, a Lei nº 14.478/2022, conhecida como Marco Legal dos Criptomoedas, define as diretrizes gerais para a prestação de serviços relacionados a ativos virtuais. Essa lei estabelece fundamentos como a proteção do consumidor, a prevenção à lavagem de dinheiro, a transparência operacional e a responsabilização das prestadoras de serviços de ativos virtuais (Virtual Asset Service Providers – VASPs). Todavia, a lei não aborda diretamente protocolos descentralizados, se referindo majoritariamente a plataformas centralizadas, como bancos, instituições financeiras e corretoras.

Nesse contexto, um marco recente de relevância internacional do *DeFi* foi a aprovação, em 2025, da *GENIUS Act (Guiding and Establishing National Innovation for U.S. Stablecoins Act)*, legislação dos EUA que estabelece um quadro regulatório para as chamadas payment stablecoins. A lei define que esse tipo de ativo digital, como instrumento resgatável por um valor monetário fixo, desenvolvido para ser utilizado como meio de pagamento, tenha emissão restrita a entidades autorizadas, as quais devem manter reservas em ativos líquidos, tais como depósitos bancários assegurados, títulos do Tesouro de curto prazo ou saldos junto ao Federal Reserve, em proporção de 1:1 em relação ao montante emitido (PILLSBURY, 2025).

Além disso, a norma impõe obrigações de divulgação periódica da composição das reservas e estabelece a prioridade dos detentores de stablecoins em casos de insolvência do emissor, ampliando a proteção ao investidor. Dessa forma, a *GENIUS ACT* representa não apenas um incentivo regulatório, mas também um

Versão Final Homologada
27/12/2025 13:45

passo decisivo na institucionalização de ativos digitais, conferindo maior segurança jurídica a investidores e empresas do setor cripto e criando um ambiente regulatório propício ao desenvolvimento de aplicações financeiras baseadas em contratos inteligentes. Apesar da GENIUS Act ser uma regulamentação aplicada especificamente aos Estados Unidos, o caráter global do mercado DeFi faz com que seus impactos e diretrizes reverberem também no Brasil, mesmo que indiretamente.

Outro fenômeno que tem atraído a atenção dos mercados e da literatura especializada é a adoção de criptoativos em tesourarias corporativas, popularmente designadas por *treasury companies*. Esse modelo de gestão de empresas tem como modelo de negócio a aquisição de ativos digitais (na maioria dos casos com *Bitcoin* e em menor grau com *Ethereum*) como parcela do caixa e das reservas, considerando esta posição como proteção contra a desvalorização da moeda fiduciária, diversificação do risco cambial e busca por uma reserva de valor alternativa. Essas aquisições, contudo, apresentam uma avaliação ambígua: por um lado são interpretadas como inovações de alocação de capital e de gestão de liquidez; por outro, expõem a empresa a elevada volatilidade de mercado, complexidades contábeis e riscos regulatórios e de governança.

Diversos exemplos corporativos ilustram essas dinâmicas. A *MicroStrategy*, sob a liderança de Michael Saylor, adotou o *Bitcoin* como principal ativo de reserva de tesouraria. Até a conclusão do ano de 2025, a empresa havia acumulado aproximadamente 660.624 BTC, com custo médio de aquisição em torno de US$ 74.702 por unidade, totalizando um investimento próximo de US$ 47,2 bilhões (Dados apresentados publicamente pela própria empresa). Esse processo de acumulação foi financiado por emissões sequenciais de ações e dívida corporativa, consolidando uma política explícita de utilização do *Bitcoin* como pilar estratégico de gestão de capital. Segundo relatórios corporativos, essa estratégia tem gerado ganhos expressivos, ainda que em grande parte não realizados, em função da valorização do criptoativo ao longo dos anos.

A *Tesla*, por sua vez, anunciou em 2021 a aquisição de aproximadamente US$ 1,50 bilhão em *Bitcoin* como parte de sua política de investimentos (BBC, 2021). Posteriormente, em 2022, a empresa realizou cerca de 75% de sua posição, levantando aproximadamente US$ 936 milhões. Estimativas recentes indicam que a Tesla mantém entre 9.720 e 11.509 BTC (NASDAQ, 2022).

A literatura e relatórios institucionais enfatizam que, embora a alocação em criptoativos possa funcionar como *hedge* em cenários específicos de inflação ou desvalorização cambial, decisões dessa natureza exigem clara governança, procedimentos de custódia robustos, e adequada divulgação financeira para mitigar riscos de mercado, contábeis e regulatórios.

Muitos dos desafios encontrados na operação e na custódia de ativos digitais, como interações entre protocolos e gestão de risco, são solucionáveis por meio de mecanismos de automação, programados para

Versão Final Homologada
27/12/2025 13:45

executar tarefas conforme parametros apresentados pelo mercado.

Este trabalho propõe o desenvolvimento de um sistema de automação financeira para operações com criptoativos e execução de contratos em blockchain, considerando a construção de software de forma a possibilitar a gestão completa do sistema.

Para a execução das operações financeiras, foram aplicadas estratégias baseadas no cruzamento de médias móveis, realizadas por meio de operações de *swap* (trocas entre ativos), com preferência por pares lastreados em dólar (*stablecoins*). Além disso, foi o desenvolvimento de um livro de ordens para corretoras descentralizadas e uma aplicação destinada ao teste de estratégias quantitativas, utilizando o comportamento histórico do mercado em intervalos arbitrários (*backtesting*). Por se tratar do ativo com maior capitalização de mercado, o *Bitcoin* foi escolhido como o criptoativo de referência para a realização dos testes e da coleta de dados.

A estratégia selecionada demonstra a aplicabilidade do sistema desenvolvido, pois o cruzamento de médias móveis é amplamente utilizado em análises quantitativas de mercados. Foram realizados testes com diferentes abordagens de cálculo de médias, incluindo médias móveis simples (SMA), médias móveis exponenciais (EMA) e médias móveis obtidas por ajustes de mínimos quadrados.

Além das médias móveis, durante os testes também foi considerada a estratégia de apenas comprar o ativo e mantê-lo (holding), uma vez que o uso do cruzamento de médias móveis pode ser considerado como possível alternativa ao *holding* tradicional.

Os resultados obtidos foram armazenados, tratados e apresentados de forma comparativa, destacando o contraste entre as características teóricas de cada tipo de média móvel e os resultados observados nos diferentes ciclos de alta e baixa do *Bitcoin*.

Para a definição de parâmetros como fundos e topos dos ciclos, recorreu-se à teoria de ciclos econômicos aplicados ao *Bitcoin*, pressupondo de que o ativo digital apresenta fases recorrentes associadas a eventos programados em seu protocolo, como o halving, que reduz a recompensa por bloco minerado e, historicamente, desencadeia movimentos de valorização seguidos por correções de mercado (HAYES, 2019; BOURI et al., 2019). A utilização dessa abordagem permitiu maior clareza e consistência na comparação entre períodos históricos do ativo.

Algumas etapas de desenvolvimento, especialmente aquelas relacionadas à segurança da informação, serão desconsideradas no escopo deste estudo, de modo a manter o foco no núcleo funcional da proposta. Assim, os principais objetivos deste trabalho são a elaboração de uma revisão bibliográfica sobre Finanças

Versão Final Homologada
27/12/2025 13:45

Descentralizadas (*DeFi*), o desenvolvimento dos softwares das aplicações em linguagem Python, em operação compatível com *DeFi*, e testados na rede de testes *Binance Testnet*, e a concepção da implementação dos modelos testados em ambientes reais de *DeFi*.

Em relação à formação do engenheiro físico, este trabalho se propõe a abordar conteúdos fundamentais das áreas de Administração, Economia, Algoritmos, Programação, Estatística e Metodologia Científica. A integração desses conhecimentos permitiu o autor ampliar sua visão crítica e desenvolver competências essenciais para atuar em contextos complexos e multidisciplinares. Dessa forma, busca-se a consolidação de saberes técnicos, e também a aplicação prática de estudos interdisciplinares, que são característicos da carreira do engenheiro físico e fundamentais para sua inserção em diferentes setores profissionais e de pesquisa. Assuntos relacionados a finanças descentralizadas ainda são pouco explorados em língua portuguesa, e é esperado que este trabalho contribua para o desenvolvimento de mais estudos relacionados ao assunto. Pela natureza multidisciplinar do engenheiro físico, este trabalho permite uma série de continuações envolvendo otimização dos parâmetros de operação, arquiteturas com segurança operacional para operações reais, desenvolvimento de hardware específico para hospedagem eficiente do sistema de automatização, dentre outros.

Versão Final Homologada
27/12/2025 13:45

2 Fundamentação Teórica

2.1 Automação financeira

O mercado financeiro é um ambiente extremamente complexo e volátil, caracterizado pela presença de uma vasta quantidade de informações, variáveis e incertezas, além de ser influenciado por diversos fatores econômicos, sociais e políticos (QIU et al., 2020). Com o desenvolvimento tecnológico, a velocidade das operações cresce de forma inversamente proporcional ao tempo disponível para a tomada de decisão. Dessa forma, o desenvolvimento de soluções que aumentem a eficiência das operações torna-se necessário para a criação de modelos competitivos. Segundo Baker e Wurgler (2007), o sentimento do investidor exerce um papel relevante nos movimentos de mercado, podendo levar a desvios significativos em relação aos fundamentos econômicos. Em momentos de otimismo exagerado, há uma tendência de sobrevalorização dos ativos, enquanto, em cenários de pessimismo, ocorre subvalorização, influenciando diretamente as decisões de compra e venda.

Conforme apresentado na reportagem Fear vs optimism: the top emotions driving retail investors' decisions, publicada em maio de 2025 pela plataforma eToro, a pesquisa realizada pelo Retail Investor Beat apresenta dados coletados em doze diferentes países, totalizando dez mil investidores entrevistados. Dentre os dados, oitenta e seis por cento dos investidores admitem que experiências pessoais e emoções influenciam suas escolhas de investimento. Segundo a estrategista de mercado global da eToro, Lale Akoner, "é interessante que duas emoções opostas – medo de perda e otimismo – sejam os principais motivadores citados pelos investidores de varejo como os que mais impactam suas decisões de investimento" (ETORO, 2025, n.p.).

A literatura em finanças comportamentais demonstra que vieses como excesso de confiança, ancoragem e aversão à perda desempenham papel central na tomada de decisão dos investidores (KAHNEMAN; TVERSKY, 1979; SHEFRIN, 2000). Esses fatores, quando aliados à volatilidade do mercado, potencializam resultados negativos e aumentam a probabilidade de erros estratégicos. Tendo em vista a mitigação de erros em momentos de fortes emoções, o desenvolvimento de mecanismos de automação financeira mostra-se como uma alternativa promissora para a redução da influência subjetiva nas decisões de investimento.

Nesse contexto, estudos recentes sobre aprendizado de máquina, inteligência artificial e automação aplicados ao mercado financeiro indicam que modelos algorítmicos podem auxiliar na identificação de padrões e na formulação de estratégias mais consistentes. Dessa maneira, busca-se diminuir a margem de erro ocasionada por vieses comportamentais, estabelecendo um processo decisório mais racional e alinhado a fundamentos

Versão Final Homologada
27/12/2025 13:45

econômicos.

Para além de suprir a necessidade de constante fluxo de operações, economizar tempo e mitigar os erros ocasionados pelos sentimentos do operador, a automação financeira se apresenta como alternativa para diversas circunstâncias, dentre elas: a melhora na tomada de decisão, redução nos custos operacionais, detecção de fraudes e escalabilidade.

Contudo, apesar de seu potencial transformador, a implementação de sistemas automatizados de negociação apresenta obstáculos significativos. Alguns dos maiores desafios para a automatização desse tipo de operação são a latência, o tempo que um sistema leva para receber dados do mercado e processá-los; falhas de no software ou na infraestrutura, levando a execução incorreta ou expondo vulnerabilidades do processo; e o sobreajuste, que ocorre quando um modelo estatístico se ajusta muito bem ao conjunto de dados anteriormente observado, mas se mostra ineficaz para prever novos resultados.(CHABOUD,2014)

Ao se modelar uma estratégia, é necessário compreender sua viabilidade prática antes de iniciar seu funcionamento. Para isso, são utilizados mecanismos de backtest. O backtest é uma metodologia que consiste em aplicar uma estratégia de negociação com dados históricos de mercado, de modo a verificar como a estratégia se comporta em diferentes cenários passados. Essa prática permite avaliar o potencial de rentabilidade do modelo, além de identificar pontos de vulnerabilidade e riscos associados. A partir dessa análise, é possível realizar ajustes nos parâmetros ou mesmo descartar estratégias inviáveis, reduzindo custos e evitando perdas financeiras em ambiente real.

Apresentando as principais aplicações de automação financeira em operação no mundo temos: a plataforma Aladdin desenvolvida pela BlackRock, maior gestora de ativos do mundo, essa plataforma atua como um sistema integrado para gestão de riscos, negociação algorítmica e compliance, sendo utilizada por mais de 200 instituições financeiras e responsável pelo monitoramento de trilhões de dólares em ativos globais.

Outro muito relevante para operações financeiras automatizadas é o algoritmo LOXM, desenvolvido pela JP Morgan. Baseado em inteligência artificial, esse sistema tem como objetivo executar ordens complexas em mercados de ações, reduzindo custos de transação e minimizando o impacto sobre os preços dos ativos.(Finance Magnates, 2017).

Tanto a J.P. Morgan quanto a BlackRock atuam como market makers. Isso significa que essas instituições assumem a função de prover liquidez ao mercado, colocando ordens de compra e venda de forma contínua. Na prática, elas garantem que sempre haja uma contraparte disponível para investidores que desejam negociar ativos, reduzindo a volatilidade e estreitando o spread (diferença entre o preço de compra e de venda).

Versão Final Homologada
27/12/2025 13:45

Porém, este não é o único tipo de empreendimento que se beneficia com modelos de automação financeira. A empresa Citadel Securities é um exemplo de uso intensivo de algoritmos de alta frequência (High-Frequency Trading – HFT). O HFT consiste em estratégias de negociação baseadas em algoritmos computacionais avançados que executam milhares de ordens em frações de segundo, aproveitando variações mínimas de preço. Essas operações têm como objetivo capturar oportunidades extremamente rápidas de arbitragem e, ao mesmo tempo, fornecer liquidez ao mercado. (Zaharudin, Young & Hsu, 2022). Suas operações são quase totalmente automatizadas, processando ordens em microssegundos para prover liquidez em diferentes bolsas de valores, consolidando-se como uma das líderes globais em negociação eletrônica.

Nos mercados financeiros tradicionais, como os representados pelas instituições apresentadas anteriormente, os ativos são geridos através de estruturas centralizadas, com intermediários financeiros, controles regulatórios rígidos e liquidez muitas vezes limitada por processos de liquidação convencionais.(AL-AMRI et al., 2019) Entretanto, desde 2017, experienciamos a adoção crescente de finanças descentralizadas (DeFi), um ecossistema baseado em *Blockchain* que possibilita a criação de serviços financeiros sem intermediários tradicionais, ao contrário das estruturas centralizadas dos mercados tradicionais, no *DeFi* a liquidez é provida pelos próprios usuários, por meio de pools de liquidez, e as regras de negociação são executadas de maneira transparente na blockchain. Esse modelo reduz custos operacionais, aumenta a acessibilidade global e fortalece a transparência das operações financeiras, embora ainda apresente desafios relacionados à escalabilidade, regulação e riscos de segurança cibernética (Schär, 2021).

No *DeFi*, o uso de automação financeira é possibilitado principalmente por contratos inteligentes (smart contracts), que executam regras pré-programadas sem a necessidade de intermediários. Essa característica viabiliza a criação de protocolos que automatizam funções centrais do sistema financeiro, como empréstimos, emissão de stablecoins, negociações em bolsas descentralizadas e estratégias de investimento.

Os processos de automação são facilmente integrados em sistemas *DeFi*, pois a natureza desses protocolos permite a interoperabilidade e a composição de serviços em blockchain. Isso possibilita que contratos inteligentes interajam entre si de maneira programável, criando mecanismos financeiros mais eficientes, como gestão de risco, desenvolvimento de estratégias quantitativas, rebalanceamento de portfólios e execução de ordens em exchanges descentralizadas.

Como este trabalho propõe o desenvolvimento de um sistema de automação financeira para uso em *DeFi*, dessa forma, nas seções seguintes serão destrinchados os temas referentes a essa tecnologia, assim como a contextualização histórica de seu desenvolvimento.

Versão Final Homologada
27/12/2025 13:45

2.2 Estratégias Quantitativas

O termo *estratégia quantitativa* refere-se a estratégias baseadas no comportamento do mercado. Uma estratégia de compra ou venda a um determinado valor pode ser considerada o exemplo mais simples de estratégia quantitativa. Conforme aumenta a complexidade dessa estratégia, maior é o conjunto de dados necessários e, por consequência, maior o gasto de processamento.

Sobre o uso de médias móveis como ferramenta de análise, segundo Brooks (2006), essa técnica, que pode ser aplicada a qualquer área do mercado financeiro, permite suavizar os dados de preço e identificar a direção geral da série estudada. Contudo, Brooks alerta que o uso isolado de médias móveis não deve ser adotado como sinal de compra ou venda, pois esses indicadores apenas apontam que algo está em andamento e requer atenção. Diferentemente do uso isolado, o uso de múltiplas médias móveis é compreendido como um mecanismo de geração de sinais de compra e venda.

A estratégia de cruzamento de médias móveis baseia-se na relação entre o comportamento de duas médias com períodos diferentes: uma mais curta (mais sensível) e outra mais longa (menos sensível).

Em relação ao comportamento das médias, tem-se que:

> "Se virmos a média rápida cruzar acima da linha de base, isso gerará um sinal de compra com essa ferramenta. Por quê? Porque as forças de demanda foram poderosas o suficiente, no período de tempo mais recente, para empurrar o preço acima da linha de base, o que é uma indicação de força crescente. Se, no entanto, observarmos a média mais rápida cair abaixo da linha de base, a implicação é que o quadro de curto prazo está tendo problemas. Deve-se presumir que o lado da oferta da equação está ganhando, e a fraqueza deve ser esperada." (BROOKS, 2006, p. 122).

Essa explicação sintetiza a lógica das estratégias de cruzamento de médias móveis: o cruzamento da média rápida acima da média lenta gera um sinal de entrada (compra), enquanto o cruzamento da média rápida abaixo da média lenta gera um sinal de saída (venda).

A Figura 1 apresenta um conjunto de dados de preço pelo gráfico de velas, e as linhas ciano e azul-escuro que representam, respectivamente, as médias móveis rápida e lenta.

Versão Final Homologada
27/12/2025 13:45



Fonte: Elaborado pelo autor (2025).

Figura 1 – Ilustração dos pontos de cruzamento entre a média móvel rápida e a média móvel lenta em um gráfico de preços.

Nos pontos indicados como 1, 2, 3 e 4, ocorre o cruzamento das médias, indicando momentos de compra ou venda. No ponto 1, a média rápida cruza abaixo da média lenta, representando um sinal de saída. No ponto 2, a média rápida supera a média lenta, indicando um sinal de recompra. Em seguida, nos pontos 3 e 4, essa mesma dinâmica se repete.

Em muitos casos a venda seguida da recompra possibilita um aumento na quantidade de ativos possuídos em relação à última saída. Dessa forma, quando isso ocorre, essa estratégia mostra-se adequada para situações de investimento a longo prazo, como o *holding*, por exemplo, pois tem como objetivo aumentar a posse de ativos ao mesmo tempo em que busca proteção contra quedas bruscas e crises de liquidez.

2.3 A origem das criptomoedas

Os conceitos de moedas digitais e sistemas de pagamento eletrônico criptografados emergiram durante a década de 1980, principalmente com a ascensão das comunidades "*Cypherpunks*". "*Cypherpunk*"refere-se à denominação usada para ativistas preocupados com o fim da liberdade e da privacidade em ambientes virtuais. "*Cypher*"significa cifra em português, remetendo à criptografia. Assim, podemos definir que um "*Cypherpunk*"é um ativista que defende o uso intensivo de criptografia forte e de tecnologias que ampliem a privacidade e a autonomia individual, como forma de provocar mudanças sociais e políticas . (RAMIRO, 2022)

Versão Final Homologada
27/12/2025 13:45

Os pilares ideológicos desse movimento podem ser sintetizados como: a privacidade por meio da criptografia, a descentralização, o anonimato ou pseudonimato e a resistência à censura. Essa filosofia, somada aos avanços tecnológicos da época, abriram espaço para questionar certos paradigmas estabelecidos em relação ao uso, distribuição e emissão de moedas.

O conceito de criptomoedas pode ser compreendido como moedas virtuais concebidas como alternativa às moedas fiduciárias tradicionais, permitindo a realização de pagamentos digitais por bens e serviços sem a necessidade de intermediários (AL-AMRI et al., 2019).

No que se refere ao desenvolvimento dessa tecnologia, as criptomoedas representam uma inovação de elevado valor agregado, capaz de questionar diretamente o monopólio governamental sobre a emissão do dinheiro (SPITZNAGEL, 2017).

Um projeto pioneiro de moeda virtual foi o DigiCash, proposto por David Chaum em 1989. Esse sistema apresentava protocolos criptografados que governavam as transações financeiras. Anos antes, Chaum já havia publicado trabalhos relacionados à criptografia aplicada às finanças, como Blind Signatures for Untraceable Payments (“Assinaturas cegas para pagamentos não rastreáveis”), em 1982.

Entretanto, o DigiCash não conseguiu resolver o problema da centralização da organização emissora, o que, juntamente com os altos custos operacionais, afastou os bancos, que seriam seus possíveis principais clientes (CARVALHO et al., 2021). Apesar de não ter se consolidado como mecanismo de uso geral, o DigiCash trouxe um componente crucial: a criptografia de chave pública e privada, princípio utilizado em projetos posteriores, incluindo o Bitcoin. A tecnologia de “assinatura cega” desenvolvida por Chaum permitiu que pagamentos eletrônicos se tornassem indetectáveis por fontes externas, gerando assim as primeiras incursões rumo às moedas virtuais contemporâneas.

Outro projeto de grande relevância para a evolução das criptomoedas foi o B-money, concebido pelo cientista da computação Wei Dai em 1998. Esse sistema apresentava a ideia de um dinheiro eletrônico anônimo e distribuído. Wei Dai descreveu em seu artigo “um esquema para um grupo de pseudônimos digitais não rastreáveis pagarem uns aos outros com dinheiro e fazerem cumprir contratos entre si sem ajuda externa”. Muitos conceitos inéditos apresentados no B-money tornaram-se comuns nas criptomoedas contemporâneas: a exigência de trabalho computacional para viabilizar a moeda digital, a remuneração pelo trabalho exercido e a verificabilidade coletiva das transações em um livro-razão (DAI, 1998).

Embora o projeto B-money nunca tenha sido implementado, o artigo de Wei Dai é citado como a primeira referência no whitepaper Bitcoin: A Peer-to-Peer Electronic Cash System, publicado por Satoshi Nakamoto

Versão Final Homologada
27/12/2025 13:45

em 2008.

Também concebido em 1998, mas apresentado ao público apenas em 2005, o Bit Gold foi desenvolvido por Nick Szabo. Assim como o B-money, tratava-se de um projeto teórico que não chegou a ser aplicado. Contudo, o Bit Gold é considerado um precursor direto do protocolo implementado por Nakamoto no Bitcoin.

Esse projeto reunia diferentes elementos criptográficos voltados à segurança e ao pseudonimato da rede, além de introduzir a mineração como mecanismo para alcançar a descentralização. Nesse mecanismo, as transações são agrupadas em blocos, e o minerador selecionado fica responsável por escrever esses dados na estrutura posteriormente nomeada de Blockchain. Entre suas inovações estavam a marcação de blocos com data e hora e o registro de títulos usando o conceito de proof of work (PoW, ou prova de trabalho).

No sistema Bit Gold, proposto por Nick Szabo, o processo de criação de unidades digitais baseava-se na resolução de desafios matemáticos do tipo *proof-of-work*, inspirados no mecanismo Hashcash. Cada participante utilizava poder computacional para encontrar um valor cuja aplicação de uma função hash resistente produzisse uma saída com determinadas propriedades difíceis de alcançar de forma aleatória. Formalmente, o objetivo era encontrar uma entrada x tal que:

$$H(x) < T$$

em que H representa uma função hash não invertível e T é um alvo (*target*) que determina o nível de dificuldade do problema (SZABO, 1998). Como a função hash se comporta de forma pseudoaleatória, o participante deve testar sucessivamente diferentes valores, pois não há forma de prever qual entrada resultará em um hash válido (BACK, 2002).

Uma vez encontrada uma solução adequada, ela é registrada juntamente com um carimbo de tempo distribuído, sem a necessidade de confiança em uma única entidade. Cada prova de trabalho gerada torna-se parte de uma cadeia pública de registros, em que a solução anterior serve como insumo para o próximo problema, formando uma sequência inforjável de títulos digitais. Esse encadeamento cronológico, aliado ao custo computacional das provas de trabalho, garante unicidade, verificabilidade e resistência à adulteração das unidades de Bit Gold.

Essas iniciativas, embora não tenham alcançado sucesso em larga escala, demonstram o percurso e a evolução das primeiras moedas virtuais, os desafios técnicos enfrentados em cada projeto e quais soluções

Versão Final Homologada
27/12/2025 13:45

encontradas para a viabilidade do dinheiro digital possibilitaram os avanços tecnológicos que hoje estão presentes no protocolo Bitcoin.

2.4 Ascenção Bitcoin e Blockchain

2.4.1 O Surgimento do Bitcoin e a Crise Financeira de 2008.

A crise financeira global de 2008, marcada pelo colapso do mercado imobiliário EUA, resultou em significativa perda de credibilidade e de confiança dos usuários em relação aos bancos e demais instituições financeiras. Nesse contexto, a atuação dos governos, enquanto detentores do monopólio de emissão e gestão da moeda de curso forçado, intensificou as preocupações em um cenário de instabilidade. As medidas de intervenção e resgate implementadas, embora necessárias para conter os efeitos da crise, evidenciaram a vulnerabilidade estrutural do sistema financeiro tradicional.

Como consequência da crise de 2008, alternativas para a manutenção e transferência de capital começaram a ser desenvolvidas, tendo como marco relevante a publicação do whitepaper de Satoshi Nakamoto, em 2008, que apresentou o *Bitcoin* como um sistema monetário descentralizado e independente de intermediários (NAKAMOTO, 2008). Essa proposta não apenas se configurou como uma resposta às fragilidades evidenciadas no sistema financeiro tradicional, mas também representou uma ruptura paradigmática em relação à centralização do poder econômico e ao modelo fiduciário vigente.

No documento intitulado Bitcoin: A Peer-to-Peer Electronic Cash System, Nakamoto descreve a criptomoeda como “uma versão puramente peer-to-peer de dinheiro eletrônico, que permitiria que pagamentos on-line fossem enviados diretamente de uma parte para outra, sem passar por uma instituição financeira

O *Bitcoin* foi classificado como uma criptomoeda por reunir as funções potenciais de uma moeda à tecnologia criptográfica, entendida como a capacidade de codificar ou cifrar informações digitais de modo que apenas aqueles que possuam a chave de decodificação possam acessá-las (TAPSCOTT; TAPSCOTT, 2016). Essa característica evidencia a principal motivação das criptomoedas: a eliminação da necessidade de confiança em instituições financeiras. Nesse modelo, a confiança, essencial às moedas fiduciárias, é substituída por lógica de programação e modelos matemáticos incorporados aos códigos das criptomoedas e aos protocolos de rede.

Segundo Franco e Bazán (2018), embora as criptomoedas compartilhem o mesmo objetivo das moedas tradicionais, diferenciam-se por deslocar o fundamento do poder: em vez de estar alicerçado na lei, passa a ser incorporado na tecnologia e na matemática. A matemática, nesse contexto, estabelece com precisão tanto a base monetária quanto os cálculos necessários para a emissão de novas unidades. Essa substituição confere

Versão Final Homologada
27/12/2025 13:45

aos participantes da rede descentralizada a responsabilidade de analisar os códigos e dados distribuídos, tornando-os os únicos responsáveis por suas próprias ações. Tal lógica reflete um dos lemas mais difundidos no ecossistema das criptomoedas: “não confie, verifique”.

Lehner, Hunzeker e Ziegler (2017) reconhecem a influência do ambiente político em que o *Bitcoin* foi inserido ao afirmar que “o white paper que lançou o *Bitcoin* em 2008, coincidentemente no auge da maior crise financeira em cinco gerações, constituiu não apenas um divisor de águas para a moeda digital, mas também uma oportunidade de introduzir um tipo mais seguro e diversificado em relação à moeda fiduciária. Em uma economia mundial totalmente conectada, representou uma primeira tentativa de uma moeda digital internacional, livremente negociada”.

2.4.2 Características Fundamentais do Bitcoin

Como apresentado anteriormente, o *Bitcoin* se baseia em funcionalidades já desenvolvidas em projetos anteriores de moedas virtuais, mas apresenta novas soluções para os problemas encontrados anteriormente. Dentre as maiores inovações, podemos citar o primeiro uso da tecnologia *Blockchain* e a aplicação do processo chamado proof of work (PoW, ou prova de trabalho), conhecido popularmente como mineração.

Podemos definir a *Blockchain* como “uma estrutura de dados distribuída e descentralizada que registra transações em blocos encadeados de forma cronológica e imutável, assegurada por mecanismos criptográficos e protocolos de consenso.” (NAKAMOTO, 2008; SWAN, 2015).

A aplicação da *Blockchain* abriu uma série de oportunidades para revolucionar os mercados financeiros existentes. Para além do Bitcoin, a tecnologia *Blockchain* apresentou possibilidades de transformação na realização de contratos, transações e registros. Mais aspectos relacionados aos processos técnicos envolvidos com *Blockchain* serão apresentados nas seções seguintes.

Como o *Bitcoin* se propõe a ser um ativo que descarta a necessidade de estruturas fiduciárias, é necessário compreender quais são as características desse ativo. Para analisar as características do Bitcoin, é preciso levar em consideração o whitepaper e o código do protocolo, chamado *Bitcoin Core*.

2.4.2.1 Escassez Programada

Inicialmente, temos a condição de escassez: a programação do *Bitcoin* limita a existência a 21 milhões de unidades de moeda; essa limitação implica um modelo deflacionário, se compararado com o modelo de emissão de moedas fiduciárias.

Versão Final Homologada
27/12/2025 13:45

2.4.2.2 Mineração, Incentivos e Dificuldade

Em sequencia, o processo pelo qual novos *Bitcoins* são introduzidos na rede e as transações são verificadas e adicionadas à *Blockchain*, é chamado mineração de Bitcoin. O processo de mineração fornece um incentivo para a participação de diversos atores, estimulando a descentralização da rede. O modelo de incentivo do *Bitcoin* remunera os mineradores por blocos validados. Existe um aspecto relacionado à dificuldade com a qual o bloco é minerado: a intenção é que a média de um bloco a cada 10 minutos se mantenha, independentemente do poder computacional disponível na rede, dessa forma, a depender do poder computacional (hashrate) disponivel na rede a dificuldade de mineração se altera. Essa mudança de dificuldade está relacionada com a possibilidade de que um ente computacionalmente muito poderoso desregule a periodicidade, ou de que a ausência dos maiores mineradores gere um congestionamento nas transações e na emissão de blocos.

No processo de prova de trabalho, os mineradores (ou validadores) investem energia elétrica e poder computacional para resolver desafios matemáticos. Ao solucionarem esses desafios, os mineradores recebem a possibilidade de ordenar as transações presentes em um conjunto, chamado de bloco, e enviá-las à blockchain. Pelo trabalho computacional exercido, os mineradores são remunerados com uma quantia pré-determinada de *Bitcoin* como incentivo à participação na rede. Esse modelo de remuneração também apresenta uma solução para a questão da emissão e distribuição de novas moedas.

A remuneração do *Bitcoin* é decrescente conforme o aumento de blocos minerados. Esse processo de diminuição do incentivo para o minerador é chamado de halving, ocorre a cada 210.000 blocos minerados e reduz a recompensa pela metade. Inicialmente, a recompensa por "minerar" um bloco era de 50 bitcoins. Em 28 de dezembro de 2012, ocorreu o primeiro halving, reduzindo a recompensa para 25 bitcoins. Até a presente data, já tivemos quatro halvings, reduzindo a recompensa atual para 3,125 bitcoins. Além do incentivo pelo bloco, o validador recebe o somatório das taxas pagas pelos usuários das transações daquele bloco. "Isso é um incentivo aos nós para apoiar a rede e fornece uma maneira inicial de colocar moedas em circulação, uma vez que não existe uma autoridade central para emiti-las. A adição estável de uma quantidade constante de novas moedas é análoga a garimpeiros despendendo recursos para colocar mais ouro em circulação. No nosso caso, tempo de CPU e eletricidade estão sendo consumidos." (NAKAMOTO, 2008).

Ao citar "emissão de novas moedas", o autor se refere especificamente à distribuição inicial, pois, como já apresentado, uma das características do *Bitcoin* é sua escassez, limitada a 21 milhões de unidades. Após o fim da disponibilidade de novos bitcoins, o processo de incentivo para os mineradores ficará restrito ao somatório das taxas contidas nas transações de cada bloco.

Versão Final Homologada
27/12/2025 13:45

2.4.2.3 Descentralização e Estrutura de Nós

A descentralização é outra característica muito importante do Bitcoin; ela é sustentada pelas estruturas independentes que compõem a rede, chamadas “nós” (nodes). Esses nós comunicam-se entre si para propagar transações e blocos, garantindo que a informação seja consistente em toda a rede. A arquitetura descentralizada do *Bitcoin* é composta por milhares de nós (computadores) espalhados globalmente, cada um mantendo uma cópia completa e atualizada da blockchain. Ao contrário dos sistemas financeiros tradicionais, que dependem de intermediários centralizados para validar e registrar transações, a rede *Bitcoin* opera de forma distribuída. Isso significa que não há uma única entidade controladora, ponto central de falha ou autoridade que possa censurar transações, congelar fundos ou manipular a oferta monetária. Um atacante interessado em comprometer a rede precisaria comprometer a maioria dos nós simultaneamente para ter sucesso, o que é extremamente difícil e custoso.

2.4.2.4 Parâmetros de Consenso

Por fim, outro pilar essencial para o funcionamento deste protocolo são os parâmetros de consenso. Como o *Bitcoin* propõe essa estrutura descentralizada, é necessário o uso de mecanismos de consenso para a manutenção da rede. Apesar de comumente a imutabilidade ser atribuída ao Bitcoin, essa característica estática poderia inviabilizar o sistema, pois, com o desenvolvimento tecnológico, estruturas de criptografia previamente seguras podem deixar de apresentar resistência. Dessa forma, o mecanismo de consenso é utilizado para tomar decisões sobre o protocolo de forma democrática e, em última instância, um usuário que preferir não acatar as atualizações de rede pode, salvo em exceções, continuar interagindo com a rede em outra versão.

Vale ressaltar um cenário crítico associado aos parâmetros de consenso: um atacante detendo mais de 50% do poder computacional da rede poderia arbitrar mudanças no protocolo, pois teria sozinho mais poder de voto do que toda a rede. Porém, para efeitos práticos, esse ataque é pouco provável, assim como o ataque por domínio de nós.

2.4.2.5 Governança e Atualizações do Protocolo

É importante destacar alguns aspectos associados à mutabilidade do protocolo bitcoin. Uma das questões que podem fragilizar o *Bitcoin* é a consideração da governança do protocolo. Mesmo com os conceitos apresentados no whitepaper e com os parâmetros estabelecidos na programação do protocolo, "situações críticas ao longo de sua trajetória exigiram alterações em suas regras operacionais. Essas mudanças podem ocorrer por meio de processos consensuais ou, na ausência deste, resultar em divergências irreconciliáveis que se materializam em versões distintas do protocolo"(NOTLAND, NOWOSTAWSKI e LI, 2023).

Versão Final Homologada
27/12/2025 13:45

Quando as modificações são aceitas pela maioria da comunidade e implementadas de forma retrocompatível, permitindo que nós que não adotem imediatamente a atualização continuem a operar na rede, caracteriza-se um processo denominado softfork.

Um exemplo relevante foi a implementação do Segregated Witness (SegWit), em 2017, ativado por meio da proposta de melhoria BIP-141. Essa atualização modificou a forma de armazenamento das assinaturas digitais nas transações, solucionando o problema da maleabilidade e aumentando a capacidade efetiva de transações por bloco sem alterar o limite nominal de tamanho. Além disso, o SegWit abriu caminho para soluções de escalabilidade em segunda camada, como a Lightning Network, que permitem o aumento do número de transações sem comprometer a segurança da rede (STANFORD BLOCKCHAIN, 2019).

Em contrapartida, quando as mudanças não são retrocompatíveis, impossibilitando o funcionamento conjunto entre versões antigas e novas do protocolo, ocorre o denominado hardfork. Nesse caso, a *Blockchain* divide-se permanentemente em duas cadeias distintas, cada uma regida por suas próprias regras de validação.

Um caso emblemático foi a criação do *Bitcoin* Cash (BCH), em agosto de 2017, decorrente do desacordo entre grupos que defendiam o aumento do tamanho do bloco para ampliar a capacidade transacional e aqueles que preferiam manter o limite de 1 MB associado a soluções complementares como o SegWit.

A ausência de consenso levou à cisão definitiva da rede e ao surgimento de uma nova criptomoeda (Harvey, 2018). Outro exemplo foi o *Bitcoin* Gold (BTG), lançado em outubro de 2017, cujo objetivo consistia em alterar o algoritmo de prova de trabalho (PoW) para torná-lo resistente a equipamentos ASIC, possibilitando que usuários com GPUs pudessem competir de forma mais equitativa no processo de mineração, o que buscava ampliar a descentralização do sistema (BITCOIN GOLD, 2017).

Esses episódios demonstram que a governança do *Bitcoin* não se limita ao código original elaborado por Nakamoto (2008), mas constitui um processo dinâmico de disputas, negociações e reconfigurações técnicas e políticas. Os softforks representam ajustes progressivos que preservam a unidade da rede, ao passo que os hardforks refletem momentos de ruptura e divergência estrutural, capazes de originar cadeias independentes (BITCOIN CORE, 2016).

Apesar das tensões geradas pelos hardforks, o *Bitcoin* demonstrou prosperidade e resiliência. Mesmo após divisões como a que originou o *Bitcoin* Cash, o BTC manteve sua liderança em valor de mercado, volume de liquidez e infraestrutura de suporte, como corretoras e carteiras digitais (Notland, Nowostawski e Li, 2023). A rede também preservou sua robustez técnica, sustentada pelo maior poder computacional agregado entre as criptomoedas, o que reforça sua segurança e confiabilidade (HARVEY, 2018). Ademais, a

Versão Final Homologada
27/12/2025 13:45

capacidade do *Bitcoin* de incorporar inovações incrementais por meio de softforks, aliada ao desenvolvimento de soluções complementares como a Lightning Network, evidencia que, mesmo diante de disputas internas e fragmentações, a versão original permaneceu como referência central no ecossistema de criptoativos.

2.4.2.6 Blockchaincomo Infraestrutura

Conforme os contextos apresentados, diferentes plataformas têm explorado a *Blockchain* como infraestrutura para serviços que vão desde empréstimos, derivativos e seguros, até a criação de stablecoins lastreadas em moedas fiduciárias. Essas inovações buscam não apenas substituir intermediários tradicionais, mas também propor novos modelos de governança, transparência e acessibilidade financeira.

Assim, enquanto o *Bitcoin* permanece como um caso paradigmático de criptomoeda, responsável por impulsionar a difusão do uso da tecnologia blockchain, outras infraestruturas, como o Ethereum, Solana, entre outras, desenvolveram-se com foco na implementação de contratos inteligentes. Os contratos inteligentes podem ser definidos como programas autoexecutáveis armazenados em uma blockchain, cuja lógica é desencadeada automaticamente quando condições previamente estabelecidas são atendidas (SCHÄR, 2021).

No contexto das finanças descentralizadas (DeFi), esses contratos desempenham papel central ao permitir a execução de operações financeiras – como empréstimos, trocas de ativos, derivativos e seguros – sem a necessidade de intermediários tradicionais, garantindo transparência, imutabilidade e auditabilidade às transações.

É de interesse do presente trabalho que, ao abordar o percurso histórico de desenvolvimento dessas tecnologias, seja estabelecida uma base sólida para a análise das aplicações e inovações mais atuais.

2.5 Contratos inteligentes

O conceito de contrato inteligente foi introduzido ao debate público em 1996 por Nick Szabo, em seu artigo Smart Contracts: Building Blocks for Digital Free Markets. Neste trabalho, o Szabo apresenta o conceito de "contrato inteligente", termo que se refere à compreensão de que esse novo formato contratual representaria uma quebra de paradigma.

Szabo (1996) define o contrato como um conjunto de promessas acordadas em um "encontro de mentes". Em sua análise, o uso de contratos ultrapassa as atividades estritamente mercadológicas e abrange diversos aspectos da vida em sociedade, como relacionamentos e confiança.

Nos contratos tradicionais, há um conjunto de promessas estabelecidas entre duas ou mais partes interessa-

Versão Final Homologada
27/12/2025 13:45

das e uma terceira parte responsável pela aplicação do acordo. Segundo o autor, "a aplicação de contratos tem sido tradicionalmente considerada uma função básica dos governos capitalistas" (SZABO, 1996).

Dessa forma, com o objetivo de eliminar a necessidade dessa terceira parte, aumentar a eficiência do processo e contribuir para os ideais cypherpunks de privacidade e autonomia individual, Szabo propõe os contratos autoexecutáveis. "A ideia básica dos contratos inteligentes é que muitos tipos de cláusulas contratuais (como penhoras, cauções, delineamento de direitos de propriedade etc.) podem ser incorporados ao hardware e ao software com os quais lidamos, de modo a tornar a violação do contrato onerosa (se desejado, às vezes proibitivamente) para o infrator." (SZABO, 1996).

Como exemplo canônico para ilustrar o funcionamento de um contrato inteligente, o autor apresenta a máquina de venda automática. Nela, "a máquina recebe moedas e, por meio de um mecanismo simples, distribui o troco e o produto de forma justa. Os contratos inteligentes vão além da máquina de venda automática, propondo a incorporação de contratos em todos os tipos de propriedade que são valiosos e controlados por meios digitais" (SZABO, 1996). Com a chegada do *Bitcoin* e a implementação da blockchain, houve a materialização de uma infraestrutura digital segura e descentralizada, capaz de garantir a imutabilidade e a transparência, viabilizando, assim, a execução de contratos inteligentes.

A aplicação e a utilização de redes descentralizadas constituem um fenômeno relativamente recente e ainda carecem de pareceres jurídicos consolidados, especialmente no que se refere aos modelos de negócios e produtos financeiros criados de forma autônoma por usuários ou por grupos empresariais participantes dessas redes. Embora exista dificuldade em legislar sobre as oportunidades emergentes do setor financeiro descentralizado, a plataforma jurídica brasileira JusBrasil apresenta, no artigo "*Smart Contracts*: o que são e como funcionam?", escrito pela advogada Dra. Viviane Resende, entendimentos jurídicos acerca dos contratos inteligentes.

Para diferenciar um contrato tradicional de um contrato inteligente, a advogada descreve: "Nos contratos tradicionais, se uma das partes não cumpre com os termos acordados, a outra parte, por meio do Judiciário, pode exigir o cumprimento do contrato. Já nos *Smart Contracts* não é necessária a presença de um terceiro para validar a operação e executar o acordo." Ainda em relação à imutabilidade dos contratos, afirma: "Os *smart contracts* se fazem cumprir por si só, pois são programados com uma série de regras e determinações, com base em 'se, então'. Em outras palavras, são impostas condições que, uma vez alcançadas, executam automaticamente uma determinada transação." (JUSBRASIL, 2025)

Nesse entendimento, o artigo apresenta uma seção intitulada "Onde Aplicar os *Smart Contracts*?", em que as principais utilizações destacadas estão associadas aos bancos, produtos financeiros e meios de

Versão Final Homologada
27/12/2025 13:45

pagamento. Contudo, também há destaque para aplicações que não envolvem a transferência de ativos, como o rastreamento da localização de alimentos em cadeias de suprimentos, permitindo o registro de dados sobre a origem dos produtos e a variação de temperatura durante o transporte. Essas aplicações, não necessariamente financeiras, compõem um uso da blockchain denominado, *Real Word Assets* (RWA). (JUSBRASIL, 2025)

As características do mercado descentralizado e a utilização de contratos inteligentes são altamente propícias à implementação de mecanismos e técnicas de automação financeira. O desenvolvimento de sistemas capazes de interagir com esse mercado, por meio de diversas redes e protocolos, tem sido empregado para maximizar posições de investimento e minimizar riscos, como crises de liquidez. Independentemente da estratégia adotada, como a *Blockchain* armazena todos os seus dados de forma pública e em tempo real, a aplicação de metodologias quantitativas possiblitam, então, ganhos significativos.

2.6 Finanças descentralizadas

Este termo se refere ao conjunto de aplicações financeiras em criptomoedas ou em blockchain, voltadas para substituir intermediários ou atravessadores financeiros, é comumente denominado Finanças Descentralizadas (*DeFi*). Considera-se parte do *DeFi* o ecossistema de aplicações, serviços e infraestruturas financeiras alternativas com o objetivo de criar um novo sistema financeiro descentralizado (SEIDEL, 2018).

A compreensão das partes que compõem o ecossistema das finanças descentralizadas exige uma delimitação clara em relação aos modelos financeiros tradicionais e centralizados que o precederam. O sistema financeiro tradicional, também conhecido como *TradFi*, é caracterizado pela centralização e pela intermediação obrigatória de instituições financeiras, como bancos, corretoras e bolsas de valores. Esse modelo é amplamente regulamentado por autoridades estatais, apresentando segurança jurídica e estabilidade institucional. (ALAMSYAH et al., 2024)

Instituições como corretoras e exchanges são responsáveis pela negociação de criptoativos, mas ainda apresentam modelos de negócio, custódia e gestão centralizados. Essas instituições são classificadas como finanças centralizadas (*CeFi*), uma vez que representam uma transição entre o sistema financeiro tradicional e o descentralizado.

Um dos benefícios das operações *CeFi* é que os usuários confiam seus fundos a uma entidade responsável pela execução das operações e pela segurança dos ativos, o que reintroduz o elemento de confiança que o sistema descentralizado busca eliminar. Em alguns casos, a custódia oferecida por esse tipo de instituição pode ser um diferencial.

Versão Final Homologada
27/12/2025 13:45

Dessa forma, observa-se que as principais diferenças entre *DeFi*, *CeFi* e *TradFi* são fundamentadas na tecnologia *Blockchain* e no uso de contratos inteligentes. (QIN et al., 2021)

Graças à estrutura fornecida pela blockchain, a transparência é garantida pela natureza pública das transações e pela imutabilidade dos registros, proporcionando um ambiente financeiro autônomo e acessível globalmente.

A transferência da estrutura para um modelo operante em *DeFi*, voltado a operações reais, necessitaria apenas da alteração no modo de interação entre o software e o sistema descentralizado, mantendo as funções lógicas e arquiteturas intactas. (NARAYANAN et al., 2016)

Uma das características da operação *DeFi* é a transparência e a distribuição pública dos dados na *blockchain*. Dessa forma, nomeamos esse conjunto de dados disponíveis, que compõem a *blockchain*, como dados *on-chain*.

Facilitados pela natureza dos mecanismos *DeFi*, esses dados podem ser utilizados para a aplicação de estratégias quantitativas de compra, venda e outras operações financeiras diversas, por meio de operações lógicas e de um sistema automatizado.

Para a automação das operações, foi necessário o desenvolvimento de mecanismos de comunicação entre a *blockchain* e as operações lógicas que regem as tomadas de decisão. Para dar início a essa interação, ainda de forma não automatizada, foi desenvolvido um mecanismo de livro de ordens para utilização em redes descentralizadas.

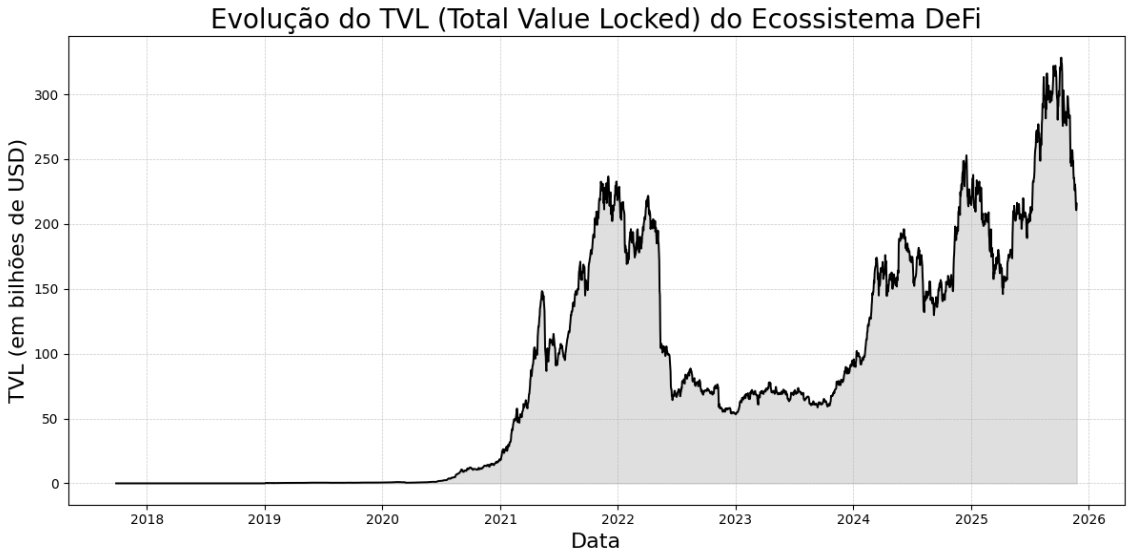
Livros de ordens são sistemas amplamente difundidos em ambientes centralizados, muitas vezes responsáveis por parcelas significativas dos lucros das instituições financeiras tradicionais. Com a característica de descentralização dos mercados *DeFi*, não há entidade capaz de gerir e garantir o livro de ordens, como ocorre nas finanças tradicionais. Dessa forma, um livro de ordens capaz de executar uma determinada operação quando um dado fenômeno ocorrer, como uma alta ou baixa de preços, torna-se uma alternativa a esse sistema em ambientes *DeFi*.

Com uma conexão capaz de realizar operações financeiras em ambientes *DeFi* e *CeFi*, foi desenvolvido um modelo de automatização de operações de compra e venda que recebe uma estratégia quantitativa expressa por meio de operações lógicas e executa as operações em tempo real.

Versão Final Homologada
27/12/2025 13:45

Atualmente, o mercado *DeFi* é composto por um montante acumulado de 155,082 bilhões de dólares (Defillama, 2025). Esse valor refere-se ao *Total Value Locked* (TVL), ou seja, o valor total de ativos digitais bloqueados em contratos inteligentes de um protocolo de finanças descentralizadas.

Normalmente, um TVL consistentemente alto sugere que um aplicativo descentralizado é robusto e estável, com forte utilidade, bem como a confiança e a credibilidade de seus usuários (Coingecko, 2025).



Fonte: Elaborado pelo autor (2025).

Figura 2 – Evolução histórica do valor total bloqueado (TVL) no ecossistema *DeFi* entre 2018 e 2025.

A Figura 2 apresenta gráfico apresenta o histórico do TVL nos últimos anos. A partir dele, podemos destacar os principais acontecimentos desse mercado e comparar a reação a esses eventos com a variação do TVL. Para a elaboração do gráfico foram requisitados os dados disponíveis na plataforma *Defillama*.

Em 2020, observamos a primeira alta expressiva: o valor total bloqueado (TVL) saltou de 1 bilhão de dólares para 15 bilhões de dólares em poucos meses. Esse fenômeno ficou conhecido como *DeFi* Summer. O evento marcou um ponto de inflexão no desenvolvimento do ecossistema blockchain, demonstrando o potencial transformador da descentralização sobre a estrutura tradicional dos mercados financeiros. Durante o *DeFi* Summer, plataformas baseadas na *Blockchain* Ethereum, como *Uniswap*, *Compound*, *Aave* e *Yearn.Finance*, ganharam destaque ao oferecer serviços financeiros sem intermediários. (STEVENS, 2020; OUTUMURO, 2020)

Versão Final Homologada
27/12/2025 13:45

Além dos mecanismos voltados para o retorno financeiro, os projetos *DeFi* apresentavam governança descentralizada por meio de tokens de governança. Dessa forma, usuários que possuíam esses tokens também participavam ativamente das decisões de desenvolvimento e atualização dos protocolos.

Em 2021, o mercado passou por um período de expansão massiva. Nesse ano, ascenderam outras *Blockchains* compatíveis com a EVM (*Ethereum Virtual Machine*), como *Binance Smart Chain* (BSC), *Polygon* e *Avalanche*, ampliando significativamente o ecossistema. Com o aumento da procura por *Bitcoin* e *Ether*, o TVL, impulsionado pelo mercado, atingiu um topo de aproximadamente US$ 230 bilhões.

No ano seguinte (2022), grande parte do crescimento obtido nos anos anteriores foi revertida por uma forte crise de confiança. Diversos fatores contribuíram para derrubar o potencial das finanças descentralizadas e minar a confiança dos usuários.

Em maio desse mesmo ano, a *stablecoin* UST, pertencente ao ecossistema *Terra* e projetada para manter paridade com o dólar norte-americano por meio de um mecanismo algorítmico de arbitragem com seu outro *token* (*LUNA*), perdeu seu lastro após uma série de saques massivos e falhas no modelo de estabilidade. O colapso do ecossistema Terra reduziu seu valor de aproximadamente US$ 60 bilhões para quase zero, evidenciando a vulnerabilidade de sistemas de *stablecoins* algorítmicas não colateralizadas.(BADEV, 2023)

Ainda em 2022, fundos de investimento e plataformas de empréstimos centralizadas, como *Celsius*, *Voyager* e *Three Arrows Capital*, enfrentaram colapsos devido à exposição excessiva a ativos ilíquidos e à alavancagem insustentável. A incapacidade de honrar retiradas e obrigações financeiras resultou em uma série de falências e reestruturações judiciais, comprometendo ainda mais a credibilidade do setor.

Aproximando-se do fim do ano, ocorreu o colapso de mais uma grande corretora de criptomoedas, a *FTX*. A empresa entrou em colapso após a revelação de práticas fraudulentas e do uso indevido de fundos de clientes pela empresa associada Alameda Research. O episódio expôs graves falhas de governança corporativa e de transparência, resultando na perda de mais de US$ 30 bilhões em valor de mercado e em investigações regulatórias globais.(REIFF, 2023)

Esses fatores críticos, somados à necessidade de correção de preços após três anos de crescimento sem precedentes, levaram a uma forte retração do mercado. Nesse ano crítico, o valor do TVL caiu para menos de 50 bilhões de dólares.

Em 2023, o mercado, já abalado pelo período anterior, passou a focar em propostas voltadas ao desenvolvimento de mecanismos de segurança para contratos inteligentes, soluções de eficiência operacional,

Versão Final Homologada
27/12/2025 13:45

minimização de taxas de transação, implementação de protocolos de RWA (*Real World Assets*) e revisão das promessas feitas em 2021, considerando o novo cenário.

Entre os anos de 2024 e 2025, o mercado *DeFi* vivenciou um período de retomada da inovação, marcado pelo surgimento de novos protocolos e pela consolidação do uso de *Blockchains* de segunda camada (*Layer 2*). Essas redes complementares tiveram papel fundamental na ampliação da escalabilidade e na redução de custos de transação, permitindo o desenvolvimento de soluções mais eficientes e acessíveis dentro do ecossistema cripto.

Ainda em 2024, produtos financeiros baseados em *staking* e *staking líquido* ganharam expressiva tração no mercado. Esses mecanismos, amplamente adotados por *Blockchains* que operam sob o modelo de prova de participação *(Proof of Stake* – PoS), têm como função primordial contribuir para a validação de blocos e segurança da rede. (GOGOL, 2024)

Na rede *Bitcoin*, conforme já mencionado, a validação das transações ocorre por meio da prova de trabalho (*Proof of Work* – PoW), um modelo que utiliza o poder computacional dos mineradores para resolver problemas criptográficos complexos. Em contrapartida, *Blockchains* que utilizam o modelo PoS substituem a mineração pela participação financeira direta dos validadores, que bloqueiam (*stake*) determinada quantidade de *tokens* nativos como forma de demonstrar comprometimento com a rede.

Nesse sistema, os validadores são selecionados de forma probabilística, de acordo com o valor e o tempo em que seus ativos permanecem bloqueados. Em troca de sua participação e da segurança oferecida à rede, esses validadores recebem recompensas periódicas, geralmente na forma de novos *tokens* emitidos pelo protocolo ou taxas de transação. Esse processo torna o *staking* uma alternativa de rendimento passivo para os investidores e um pilar essencial da governança e estabilidade das *Blockchains* PoS.

O *staking* líquido (*liquid staking*), por sua vez, representa uma evolução desse conceito. Ele permite que o usuário mantenha liquidez sobre os ativos bloqueados ao receber um token derivado que representa sua participação em staking. Esse token pode ser livremente negociado, utilizado em outras aplicações *DeFi* ou empregado como garantia em protocolos de empréstimo e liquidez. (GOGOL, 2024)

Ainda em 2024, foi observada uma crescente ascensão de protocolos *cross-chain*, esses protocolos resolviam a problemática da fragmentação da liquidez em diversas redes blockchain, pois com o aumento da capacidade de interoperabilidade, entre as redes do mercado pelo protocolo *cross-chain*, muitas combinações ascenderam para o horizonte de possibilidades dos usuários de *DeFi*.

Versão Final Homologada
27/12/2025 13:45

No ano de 2025, voltamos a patamares de ganhos expressivos chegando próximo ao valor de alta histórica do ano de 2021. Neste ano, grandes movimentos institucionais para a adoção de criptomoedas, para composição de caixa e tesouro das empresas, regulamentação amigável para empreendedores do meio cripto e perspectiva do corte de juros pelo *Federal Reserves* (FED), Banco Central dos EUA.

Em relação ao corte da taxa de juros, é esperado que o corte tende a estimular o crescimento do mercado *DeFi* por diversos canais econômicos. Quando os juros caem, o custo de oportunidade de manter capital em instrumentos de renda fixa tradicionais diminui, o que leva investidores a buscar alternativas de maior rentabilidade, como cripto ativos e produtos *DeFi*.(WU, 2025)

Historicamente, momentos em que há cortes na taxa de juros, têm sido associados à expansão nos mercados cripto, podemos usar como exemplo o ocorrido entre 2020 e 2021, quando políticas monetárias mais flexíveis coincidiram com o boom inicial das finanças descentralizadas. (POLIZU et al., 2023)

Como podemos observar a partir do histórico recente do *DeFi*, a ascensão dessa modalidade financeira tem ganhado cada vez mais utilidade e preferência, seja pela maior eficiência dos sistemas descentralizados, seja pelo crescente descrédito das instituições que detêm o monopólio da emissão do dinheiro fiduciário.

Entre os principais produtos do ecossistema *DeFi*, destacam-se as stablecoins, criptoativos desenvolvidos com o objetivo de manter seu valor estável em relação a uma moeda de referência, geralmente o dólar norte-americano (USD).

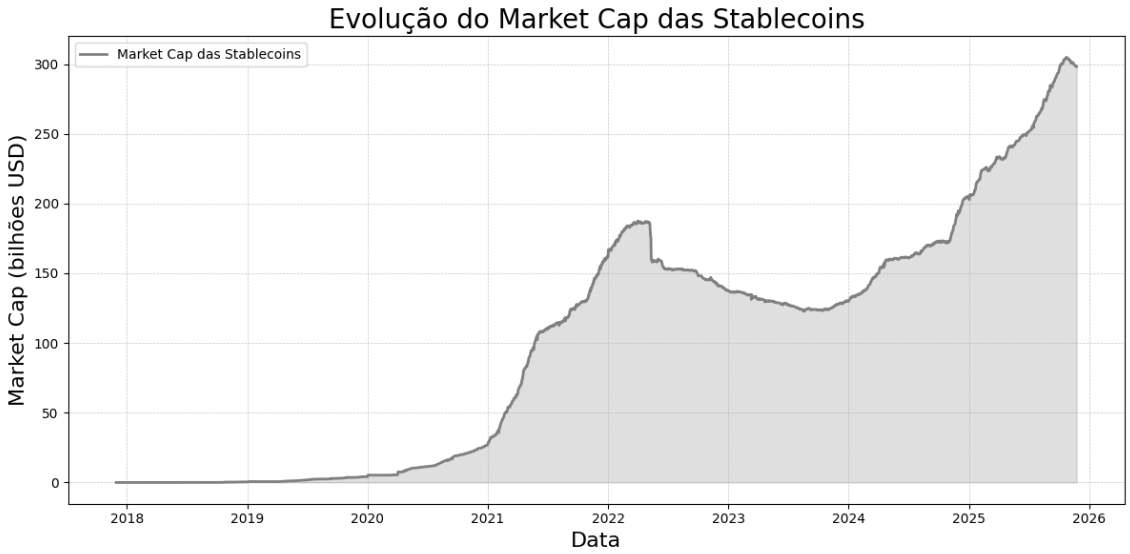
As stablecoins lastreadas em dólar podem operar por meio de diferentes mecanismos de paridade. Entre as mais tradicionais estão as fiat-backed stablecoins, como *USDT (Tether)* e *USDC (USD Coin)*, que mantêm reservas equivalentes em moeda fiduciária depositadas em instituições financeiras.(WEN et al., 2025)

Outro modelo de funcionamento é o das stablecoins colateralizadas por criptoativos, como a *DAI*, emitida pelo protocolo *MakerDAO*. Nesse caso, o valor da stablecoin é garantido por depósitos de outras criptomoedas (como *Ethereum*) em contratos inteligentes. Esses contratos bloqueiam o colateral e ajustam automaticamente as posições para manter a estabilidade do preço.

Caso haja interesse em emitir uma stablecoin sem reservas centralizadas, existe o modelo das stablecoins algorítmicas, que utilizam mecanismos automáticos de emissão e queima de *tokens* para equilibrar a oferta e a demanda. (MITA et al., 2025)

Versão Final Homologada
27/12/2025 13:45

Independentemente do protocolo utilizado para manter a paridade com o dólar, as *stablecoins* são de extrema importância para o funcionamento dos mecanismos do *DeFi*. Ao oferecerem uma alternativa à volatilidade dos preços dos *tokens* e uma paridade com a moeda fiduciária, permitem que os usuários repliquem ou desenvolvam novas operações financeiras, beneficiando-se da maior eficiência proporcionada pelos mercados descentralizados.



Fonte: Elaborado pelo autor (2025).

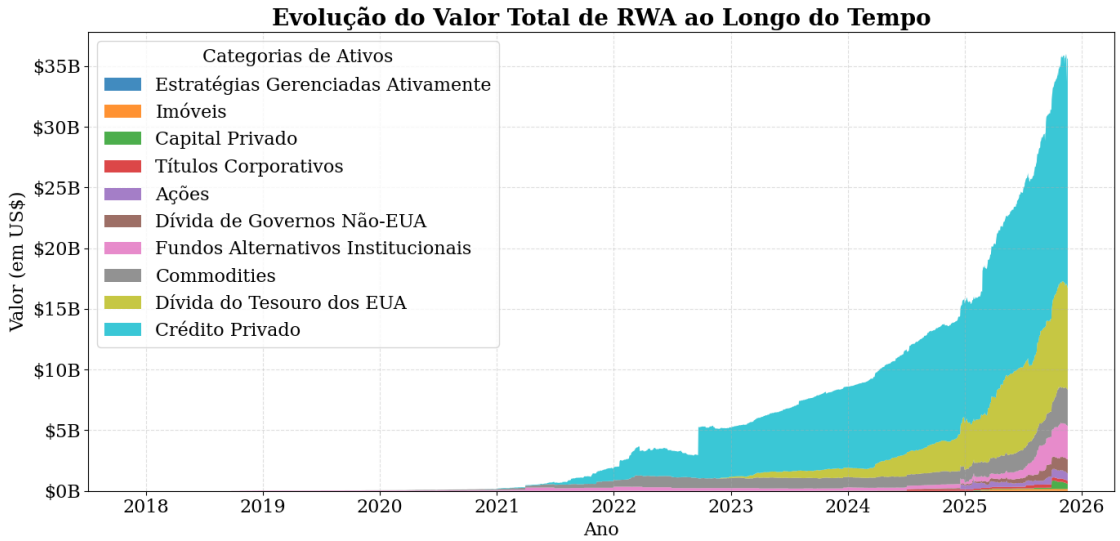
Figura 3 – Evolução do valor de mercado (market cap) das stablecoins entre 2018 e 2025.

De toda forma, o mercado de *stablecoins* conta com uma capitalização total de aproximadamente US$ 308,477 bilhões, considerando tanto os tokens bloqueados em protocolos quanto aqueles disponíveis para livre movimentação (DefiLlama, 2025).

Como podemos observar na Figura 3, no mesmo período em que ocorreram grandes oscilações no TVL total do mercado, a capitalização das stablecoins apresentou relativa estabilidade, demonstrando seu papel como reserva de valor e instrumento de liquidez dentro do ecossistema *DeFi*.

Outra modalidade que vem crescendo exponencialmente é a dos ativos do mundo real tokenizados, conhecidos como *Real World Assets* (RWA). Essa modalidade representa ativos do mundo real, tangíveis ou não, que passam a estar disponíveis para negociação por meio do mercado cripto e do ecossistema *DeFi*.

Versão Final Homologada
27/12/2025 13:45



Fonte: Elaborado pelo autor (2025).

Figura 4 – Evolução do valor total tokenizado em ativos do mundo real (RWA) por categoria, entre 2018 e 2025.

Como podemos observar no gráfico apresentado na Figura 4, há um crescimento expressivo dessa fatia do mercado ao longo dos últimos anos. Considerando o mercado total de RWAs, atualmente há uma capitalização de aproximadamente US$ 35,11 bilhões (RWA.xyz, 2025).

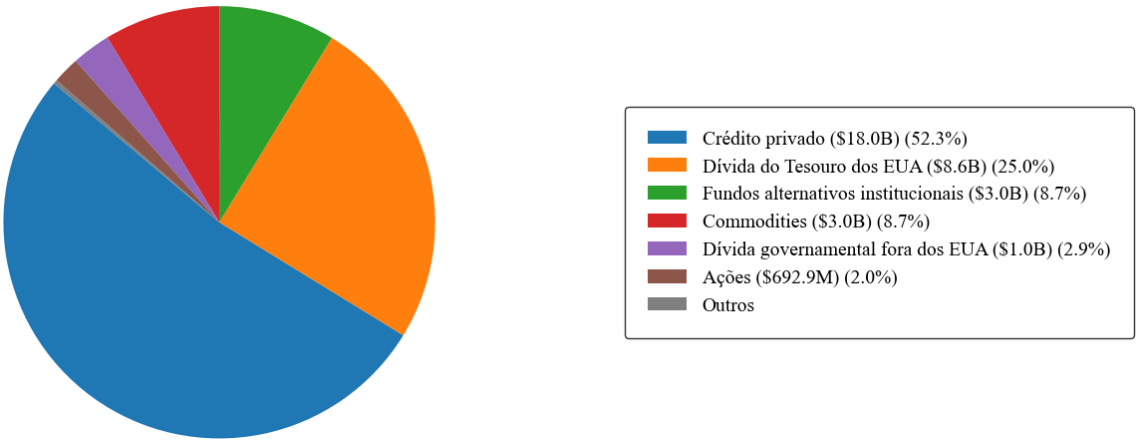
Ainda no mesmo gráfico, é possível observar o crescimento segmentado por tipo de ativo ofertado. Em ordem decrescente de capitalização, destacam-se: crédito privado, dívida do Tesouro dos Estados Unidos, *commodities*, fundos alternativos institucionais, dívida governamental fora dos Estados Unidos, ações, capital privado, títulos corporativos e estratégias gerenciadas ativamente.

Para visualizar a distribuição atual dos recursos entre as diferentes categorias de RWAs, podemos consultar o gráfico apresentado na Figura 5. Essa visualização evidencia que, atualmente, a maior parte dos ativos que compõem esse segmento do mercado é representada pelo crédito privado.

A tokenização de crédito privado consiste em converter direitos de crédito em *tokens* digitais registrados em uma blockchain. Cada *token* representa uma fração de um ativo real, permitindo que ele seja negociado de forma mais ágil, transparente e acessível.

Versão Final Homologada
27/12/2025 13:45

Esses *tokens* são adquiridos por investidores, que passam a receber os rendimentos provenientes dos pagamentos dos devedores, conforme previsto nos contratos originais. Na prática, uma instituição originadora — como uma empresa de crédito, por exemplo — emite *tokens* correspondentes a parcelas de um portfólio de créditos privados.

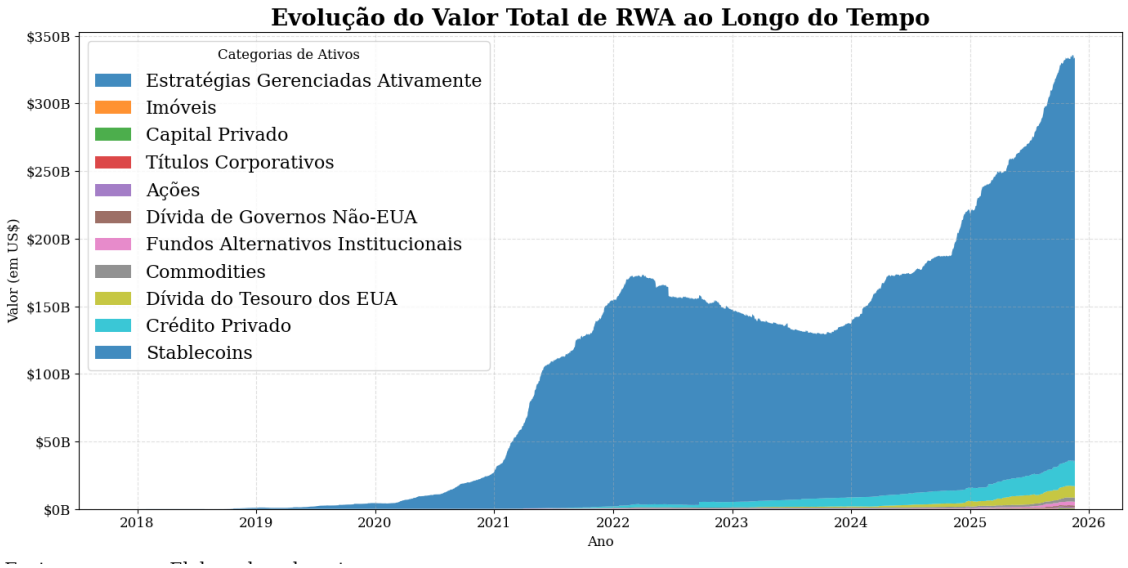


Fonte: Elaborado pelo autor (2025).

Figura 5 – Distribuição percentual do valor total alocado em diferentes categorias de ativos tokenizados.

Ao tratarmos o conceito de RWA de forma mais ampla, é possível considerar as stablecoins como parte desse mercado, uma vez que, ao emitir um token com paridade de 1:1, utiliza-se um ativo real, nesse caso o o dólar, para gerar um token que representará essa unidade on-chain (dentro da rede blockchain). Dessa forma, ao observar a Figura 6 a seguir, é possível visualizar o crescimento do mercado de RWAs nos anos anteriores, somado ao desenvolvimento do mercado de stablecoins.

Versão Final Homologada
27/12/2025 13:45



Fonte: Elaborado pelo autor (2025).

Figura 6 – Evolução temporal do valor total (market cap) de ativos do mercado RWA, segmentados por categoria.

Segundo as estimativas apresentadas neste ano de 2025 pelos pesquisadores da Animoca Brands, Andrew Ho e Ming Ruan, “o cenário atual revela um ecossistema em rápida expansão”. Ainda neste artigo, os autores ressaltam a possibilidade de aumento da capitalização de mercado do segmento de RWA, em um cenário otimista, passando do patamar atual de 35 bilhões de dólares para mais de 400 trilhões de dólares (HO; RUAN, 2025). Outra previsão, realizada pelo *Boston Consulting Group* (BCG) em parceria com a *Ripple*, empresa de infraestrutura de ativos digitais focada em pagamentos, apresenta que “no geral, espera-se que os ativos tokenizados aumentem para mais de US$ 18 trilhões, refletindo uma taxa de crescimento anual composta de 53% até 2033” (Ripple, 2025).

Tendo em vista as estimativas de expansão do cenário atual do *DeFi*, observam-se amplas oportunidades de desenvolvimento tecnológico nesse meio, configurando-se um espaço fértil para inovação e crescimento.

Versão Final Homologada
27/12/2025 13:45

3 Metodologia

Para o desenvolvimento do sistema proposto, foram projetados três mecanismos, considerando a interação com a rede, a coleta e o tratamento dos dados. Esses mecanismos apresentam-se da seguinte forma:

- ***Backtest:*** mecanismo para simulação da aplicação de estratégias quantitativas em períodos históricos passados.
- ***Livro de Ordens:*** mecanismo para agendamento de compras e vendas conforme parâmetros previamente estabelecidos.
- ***SwapBot:*** mecanismo utilizado para aplicar as estratégias quantitativas em mercados em operação real.

Como proposta para o desenvolvimento de alternativas aos mecanismos tradicionais de automação financeira, este trabalho apresenta a construção de soluções voltadas à operação de criptoativos e contratos inteligentes em *blockchain*.

Todas as ferramentas desenvolvidas mostram-se adequadas e compatíveis com o uso de redes e infraestruturas *DeFi*. Contudo, considerando o escopo do trabalho, que visa demonstrar a viabilidade do uso de mecanismos de automação para a aplicação de estratégias quantitativas, as validações das estratégias e das operações em tempo real foram realizadas na *Binance Testnet*, uma rede de testes oferecida e hospedada de forma centralizada pela *Binance*.

Esse sistema automatizado é chamado de *SwapBot*, pois a operação *swap*, em inglês, significa “troca” e, ao falar de operações de compra e venda, estamos, na verdade, tratando de trocas entre os ativos *Bitcoin* (BTC) e dólar sintético (USDC, USDT, entre outros).

Para validar a efetividade dessas estratégias quantitativas, foi desenvolvida uma estrutura para a realização de simulações em intervalos arbitrários do passado, com o intuito de averiguar a performance, supondo a atuação nesse período de tempo. Essa estrutura também pode ser chamada de *backtest*.

A estratégia escolhida para validar o modelo de operação e demonstrar as motivações para o desenvolvimento de tecnologias financeiras voltadas ao mercado das finanças descentralizadas foi a estratégia do cruzamento de médias móveis. Como apresentado na revisão bibliográfica, essa estratégia baseia-se no

Versão Final Homologada
27/12/2025 13:45

cruzamento de duas médias calculadas a partir de conjuntos de dados distintos: uma menor (referente à média rápida) e outra maior (referente à média lenta).

3.1 Cálculo de médias móveis

O cálculo de médias móveis pode ser realizado de diversas maneiras. Para demonstrar um uso real deste mecanismo, propôs-se, além de apresentar a performance da estratégia, o teste simultâneo apresentando variações no cálculo da média, com os resultados referentes ao uso de médias móveis simples (SMA), médias móveis com ajuste exponencial (EMA) e médias móveis de mínimos quadrados com aproximação linear (LSMA).

3.1.1 Médias móveis simples (SMA)

$$SMA_k = \frac{1}{k} \sum_{i=n-k+1}^{n} p_i \tag{3.1}$$

A Equação (3.1) representa o cálculo da Média Móvel Simples (SMA). A seguir, são descritas as variáveis que compõem a equação:

- SMA_k: valor da média móvel simples calculada com uma janela de tamanho k;
- k: número de períodos considerados no cálculo da média;
- p_i: valor do preço (ou outro indicador) no instante i;
- n: índice do último período considerado na série;
- $\sum_{i=n-k+1}^{n}$: somatório dos últimos k valores da série temporal.

A fórmula indica que, para cada ponto no tempo n, são somados os k valores mais recentes, mantendo todos com o mesmo peso. O parâmetro k é responsável por indicar a sensibilidade da média: valores baixos de k tornam a média mais sensível a flutuações, enquanto valores maiores associados a k produzem uma média mais suavizada.

3.1.2 Médias móveis com ajuste exponencial (EMA)

$$EMA_t = \frac{x_t + (1-\alpha)x_{t-1} + (1-\alpha)^2 x_{t-2} + \ldots + (1-\alpha)^t x_0}{1 + (1-\alpha) + (1-\alpha)^2 + \ldots + (1-\alpha)^t}$$

Versão Final Homologada
27/12/2025 13:45

$\alpha = \frac{2}{t+1}$

(3.2)

A Equação (3.2) representa o cálculo da Média Móvel com Ajuste Exponencial (EMA). A seguir, são descritas as variáveis que compõem a equação:

- x_t: valor observado no instante t;
- x_{t-1}: valor observado no instante anterior;
- x_0: primeira observação da série temporal;
- α: fator de suavização;
- EMA_t: média móvel exponencial no instante t.

O fator α, com $0 < \alpha \leq 1$, determina a rapidez com que os valores antigos perdem influência. Assim, quanto maior o valor de α, maior o peso atribuído às observações recentes e mais sensível é a média a novas variações na série. O uso da EMA é especialmente útil para detectar tendências e mudanças de comportamento em séries com maior volatilidade.

3.1.3 Médias móveis de mínimos quadrados com aproximação linear (LSMA)

$$a = \frac{\sum XY \cdot \sum X - \sum Y \cdot \sum X^2}{(\sum X)^2 - n(\sum X^2)} \tag{3.3}$$

$$b = \frac{\sum X \cdot \sum Y - n \sum XY}{(\sum X)^2 - n(\sum X^2)} \tag{3.4}$$

Assim, a equação da reta de regressão é:

$$Y = a + bX \tag{3.5}$$

onde:

Versão Final Homologada
27/12/2025 13:45

- Y: variável dependente;
- X: variável independente;
- a: intercepto da reta de regressão;
- b: coeficiente angular;
- n: número de observações da amostra.

Este tipo de média calcula uma linha de tendência linear ajustada aos últimos n pontos de dados, utilizando o método dos mínimos quadrados, onde a consiste no ponto intercepto da reta de regressão e b o coeficiente angular da reta de regressão.

3.2 Arquitetura, Lógica e Mecanismo de Tomada de Decisão do Sistema

3.2.1 Arquitetura e funcionamento

Para o desenvolvimento dos códigos, foi utilizada a linguagem *Python*, uma linguagem de alto nível amplamente empregada no contexto científico. Para a apresentação dos dados, utilizou-se programação em *HTML (HyperText Markup Language)*, linguagem de marcação padrão para a construção da estrutura de páginas web, empregada na criação de uma *dashboard*.

Na construção das soluções apresentadas, foram utilizadas bibliotecas de programação em *Python*. Essas bibliotecas permitem o uso de funções, classes e rotinas pré-desenvolvidas, otimizando o processo de desenvolvimento e garantindo maior eficiência na implementação dos algoritmos.

Inicialmente, para o armazenamento dos arquivos, foi utilizado o formato *CSV (Comma-Separated Values)*. Esse tipo de arquivo é amplamente utilizado no armazenamento e troca de dados tabulares, como planilhas e tabelas de bancos de dados. Nesse formato, cada linha representa um registro, enquanto cada coluna, separada por vírgulas, apresenta um parâmetro desse registro.

Com o avanço do desenvolvimento e a maior complexidade no tratamento dos dados, optou-se pela utilização do formato *JSON (JavaScript Object Notation)* para os resultados consolidados, mantendo os arquivos com os dados detalhados de cada cenário calculado em *CSV*.

Nesse formato, o usuário pode utilizar os dados de cada estratégia em análises individuais, ao mesmo tempo em que dispõe de um conjunto unificado para comparação de performance.

Versão Final Homologada
27/12/2025 13:45

Para realizar a análise de performance entre as diferentes médias testadas, foram selecionados períodos de análise baseados nos ciclos de alta e baixa do Bitcoin. As performances foram testadas nos seguintes cenários: Fundo-Fundo (FF), Fundo-Topo (FT), Topo-Topo (TT)e Topo-Fundo (TF). As datas correspondentes a esses cenários são:

- Fundo-Fundo: 18/11/2011 até 14/01/2015 ; 14/01/2015 até 15/12/2018; e 15/12/2018 até 21/11/2022.
- Fundo-Topo: 14/01/2015 até 17/12/2017; 15/12/2018 até 10/11/2021; e 21/11/2022 até 22/07/2025.
- Topo-Topo: 30/11/2013 até 17/12/2017; 17/12/2017 até 10/11/2021; e 10/11/2021 até 22/07/2025.
- Topo-Fundo: 30/11/2013 até 14/01/2015; 17/12/2017 até 15/12/2018; e 10/11/2021 até 21/11/2022.

O intervalo compreendido entre 18 de novembro de 2011 e 14 de janeiro de 2015 corresponde a uma fase inicial do *Bitcoin*, marcada por baixa liquidez, escassez de dados disponíveis e assimetrias de informação. Dessa forma, optou-se pela não utilização desse período para análise, uma vez que os mecanismos de precificação ainda não refletiam a dinâmica de um ativo amplamente negociado.

Dessa forma, os períodos referentes aos primeiros Fundo-Fundo, Topo-Topo e Topo-Fundo não serão considerados na análise.

Para a construção das séries temporais utilizadas nas simulações, foram adotadas velas (*candlesticks*) com periodicidade diária, essa escolha deve-se ao fato de que velas de 1 dia representam valores médios estáveis para a geração de sinais consistentes, evitando oscilações excessivas que poderiam ocorrer em períodos muito curtos. Assim, os resultados obtidos representam o comportamento macro do ativo ao longo dos ciclos analisados.

Para a construção da parte operacional do sistema, composta pela *Lista de Ordens*, pelo *SwapBot* e pelo módulo de *Backtest*, foram utilizadas as seguintes bibliotecas da linguagem *Python*:

- *os*: utilizada para interações com o sistema operacional, como manipulação de diretórios e arquivos;
- time: fornece funções para controle de tempo e pausas de execução;
- *datetime*: utilizada para manipulação de datas e horários, essencial na análise temporal de dados financeiros;
- *json*: utilizada para leitura e escrita de arquivos no formato *JSON*;
- *threading*: permite a execução de múltiplas tarefas simultâneas por meio de *threads*;

Versão Final Homologada
27/12/2025 13:45

- *queue*: utilizada para gerenciamento de filas de tarefas e comunicação entre *threads*;
- *tkinter*: biblioteca padrão do *Python* para criação de interfaces gráficas (*GUI*);
- *numpy*: utilizada para manipulação de vetores, matrizes e operações matemáticas de alto desempenho;
- *pandas*: empregada para análise e manipulação de dados em estruturas tabulares;
- *matplotlib*: utilizada para geração de gráficos e visualização de dados;
- *python-binance*: biblioteca que permite a interação com a *API* da corretora *Binance*, possibilitando operações em ambiente *CeFi*;
- *yfinance*: utilizada para coleta de dados históricos de preços de ativos financeiros;
- *mplfinance*: biblioteca voltada para visualização de gráficos financeiros, como candles e volumes de negociação.

3.2.2 Lógica Operacional e Processo Decisório

3.2.2.1 Metodologia de Operação do Algoritmo

O algoritmo para automação das negociações foi desenvolvido em três etapas principais: obtenção de dados, atualização de indicadores e aplicação da lógica decisória. Ao iniciar o processo, define-se o estado inicial do sistema, atribuindo à variável `in_position` o valor falso caso não haja uma posição ativa no mercado, e verdadeiro caso contrário.

3.2.2.2 Etapa 1 – Coleta dos Dados de Mercado

Para a execução do programa, faz-se necessária a requisição periódica dos preços atuais do ativo por meio da API da *Binance*. Para a realização dessas coletas, a função `get_current_price` realiza as requisições conforme os seguintes parâmetros: `Symbol`, referente ao par de ativos a ser operado, no caso `BTCUSDT`; e a periodicidade das coletas, a qual define a frequência das requisições.

Caso a função `get_current_price` retorne um valor nulo, o sistema registra o erro, aguarda 10 segundos e retorna à próxima iteração. Dessa forma, caso a rede de operações fique temporariamente inoperante, quando restabelecida, o sistema volta automaticamente a operar.

3.2.2.3 Etapa 2 – Atualização dos Dados e Indicadores

Tanto para o mecanismo de Livro de Ordens quanto para a aplicação da estratégia de cruzamento de médias móveis, conforme ocorre a coleta de novos dados, faz-se necessária a atualização dos parâmetros quantitativos avaliados. Considerando a interface de usuário, além da atualização dos parâmetros, torna-se imprescindível atualizar o gráfico exibido e os registros de log. Para a realização dessas atualizações, que ocorrem durante a execução do programa, foi desenvolvida a função `update_data()`. Essa função coleta

Versão Final Homologada
27/12/2025 13:45

os valores referentes aos fechamentos das velas que compõem as médias móveis rápida e lenta, atualiza a base local de arquivos com os novos valores obtidos, calcula as novas médias móveis e atualiza a interface de usuário, o gráfico e os *logs*.

Esse procedimento assegura que os cálculos sejam realizados sobre uma base coerente e atualizada, evitando inconsistências na lógica de detecção de cruzamentos.

3.2.2.4 Etapa 3 – Aplicação da Estratégia e Tomada de Decisão

Após as etapas anteriores, o programa executa o processo de tomada de decisão conforme a estratégia definida pelo usuário. No caso da aplicação da estratégia de cruzamento de médias móveis, apresentam-se a seguir os pseudocódigos que ilustram os processos de comparação:

```
SE (MA_rapida_anterior < MA_lenta_anterior) E
 (MA_rapida_atual > MA_lenta_atual) E
 (in_position = False):
  ENTAO executar ordem de COMPRA
  Atribuir in_position = True
```

```
SE (MA_rapida_anterior > MA_lenta_anterior) E
 (MA_rapida_atual < MA_lenta_atual) E
 (in_position = True):
  ENTAO executar ordem de VENDA
  Atribuir in_position = False
```

Fonte: Elaborado pelo autor (2025).

Figura 7 – Estrutura lógica utilizada para identificar eventos de cruzamento entre médias móveis rápida e lenta.

O primeiro algoritmo apresentado na Figura 7 demonstra o processo de comparação para a execução de uma ordem de compra, enquanto o segundo representa o processo para emissão de uma ordem de venda. Durante as comparações, o programa avalia os valores atuais das médias móveis em relação aos valores calculados no instante imediatamente anterior e verifica a posição do usuário no mercado. Quando todas as condições são satisfeitas, a ordem é executada e a variável de posição é atualizada.

As comparações realizadas pelo mecanismo de Livro de Ordens podem ser representadas conforme o seguinte pseudocódigo:

Versão Final Homologada
27/12/2025 13:45

```
SE (preco_de_venda_do_usuario < preco_atual):
  ENTAO executar ordem de VENDA
  Remover ordem do Livro de Ordens
```

```
SE (preco_de_compra_do_usuario > preco_atual):
  ENTAO executar ordem de COMPRA
  Remover ordem do Livro de Ordens
```

Fonte: Elaborado pelo autor (2025).

Figura 8 – Estrutura lógica utilizada para identificar oportunidades de operações agendadas no Livro de Ordens.

Observa-se que, para o Livro de Ordens, a comparação é realizada com base em um parâmetro de preço definido pelo usuário. Conforme o sistema coleta periodicamente os dados, ele realiza as comparações com os valores definidos. Havendo um momento oportuno, com as condições satisfeitas, a operação é executada de forma automática.

Foi de interesse deste trabalho a construção de interfaces gráficas, de modo que a operação dos *softwares* desenvolvidos pudesse ser realizada por uma maior gama de usuários, incluindo aqueles não familiarizados com programação ou com a execução de código via terminal.

3.3 Backtest

3.3.1 Descrição

O objetivo do sistema é permitir a simulação, análise e comparação de diferentes estratégias quantitativas utilizando dados históricos de mercado.

Podemos descrever suas funcionalidades como:

1. Realizar o download automático de dados históricos;
2. Executar os cálculos necessários para a estratégia escolhida;
3. Detectar automaticamente sinais de compra e venda, permitindo a automatização.

O sistema também contempla a geração automática de dashboards interativos em formato HTML, permitindo a visualização e análise dinâmica dos resultados, possibilitando a comparação simultânea entre múltiplas

Versão Final Homologada
27/12/2025 13:45

estratégias e servindo como base para a avaliação de desempenho.

Para que a simulação apresente dados em conformidade com a experiência real de um usuário naquele período, o sistema desenvolvido é capaz de considerar custos de transação (*fees*) e deslizamento (*slippage*).

3.3.2 Aspectos Técnicos do Desenvolvimento

A estrutura de arquivos fornecida pelo sistema após cada teste apresenta os seguintes elementos:

```
bcktst_YYYYMMDD_HHMMSS/ (diretório de resultados)
├─ all_backtest_results.json
├─ SMA_acumulo_tokens_trades.csv
├─ SMA_entradas_saidas_simplificado.csv
├─ EMA_acumulo_tokens_trades.csv
├─ EMA_entradas_saidas_simplificado.csv
├─ LSMA_acumulo_tokens_trades.csv
├─ LSMA_entradas_saidas_simplificado.csv
├─ dashboard_SMA.html
├─ dashboard_EMA.html
├─ dashboard_LSMA.html
├─ dashboard_unificado.html
└─ comparacao_estrategias.png
```

Fonte: Elaborado pelo autor (2025).

Figura 9 – Estrutura de arquivos gerada após a execução dos testes de *backtest*.

Os arquivos resultantes dos testes são apresentados, na Figura 9, em uma pasta que apresenta em sua nomeação um *timestamp* (registro temporal que indica o momento exato de execução do teste), garantindo a organização dos resultados.

Dentro dessa pasta, os arquivos `dashboard_SMA.html`, `dashboard_EMA.html`, `dashboard_LSMA.html` e `dashboard_unificado.html` representam, respectivamente, as dashboards específicas de cada tipo

Versão Final Homologada
27/12/2025 13:45

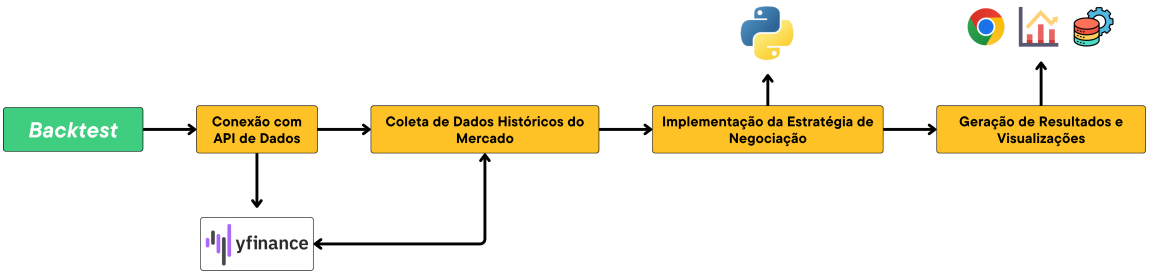
de média utilizada e a dashboard unificada.

O arquivo `all_backtest_results.json` contempla todos os dados obtidos durante a execução do teste, utilizando a arquitetura de arquivos *JSON*.

Os arquivos no formato *CSV* apresentam conjuntos de resultados mais específicos de cada teste, sendo úteis para análises individuais.

O arquivo `comparacao_estrategias.png` apresenta a comparação gráfica da performance de todas as estratégias propostas em relação à estratégia de *Holding*.

Para fins de registro e análise posterior, os resultados são exportados em múltiplos formatos, como arquivos *CSV* (com dados detalhados das operações) e *JSON* (com resultados consolidados). O sistema também gera gráficos comparativos em formato *PNG*, elaborados por meio da biblioteca Matplotlib, fornecendo uma visualização clara da evolução das estratégias ao longo do tempo.



Fonte: Elaborado pelo autor (2025).

Figura 10 – Arquitetura operacional do sistema de backtest.

Conforme ilustrado na Figura 10, o funcionamento da plataforma de backtest pode ser descrito como segue: inicialmente, uma conexão via API com o *yfinance*, responsável por fornecer os dados históricos, realiza uma requisição conforme o período e o par de ativos selecionados. Em seguida, as estratégias de negociação são implementadas e, ao final do processo, é gerada a *dashboard* contendo os dados sintetizados e gráficos disponíveis, permitindo a comparação de rendimento entre estratégias e o armazenamento organizado dos resultados localmente.

Em síntese, o sistema proposto constitui uma ferramenta para experimentação e avaliação de estratégias quantitativas no contexto de criptoativos e finanças descentralizadas, oferecendo suporte analítico e visual à

Versão Final Homologada
27/12/2025 13:45

tomada de decisões baseadas em dados.

3.4 SwapBot

3.4.1 Descrição

O sistema apresenta a execução de operações automatizadas com interface gráfica. Para realizar a comunicação entre o *software* e a execução das ordens foi utilizada a rede de testes centralizada *Binance Testnet*. Para execução das funcionalidades do *software* em aplicações reais de *DeFi*, seria necessária apenas a alteração na estrutura de comunicação, adaptando-a para interagir com redes descentralizadas.

O *SwapBot* se apresenta como uma solução para a execução automática de estratégias quantitativas, bastando ao usuário estabelecer os parâmetros e as operações lógicas desejadas. Conforme as condições são satisfeitas, as operações são executadas automaticamente.

A construção do sistema conta com uma estrutura modular que permite manutenções e atualizações em processos específicos, garantindo maior flexibilidade e escalabilidade. As camadas que compõem o sistema são:

1. Camada de Comunicação com API;
2. Camada de Lógica de Trading;
3. Camada de Interface;
4. Camada de Processamento de Dados;
5. Camada de Visualização.

Espera-se que o sistema atue de forma eficiente na execução de estratégias automatizadas, reduzindo o impacto de fatores emocionais e garantindo maior agilidade nas tomadas de decisão. O modelo modular proposto facilita a adaptação do código para novas estratégias e mercados, possibilitando escalabilidade e integração com protocolos descentralizados.

3.4.2 Aspectos Técnicos do Desenvolvimento

Para a execução das operações e comunicação com a API, foi necessário configurar o seguinte conjunto de variáveis:

- API_KEY / SECRET_KEY: Chaves de autenticação da API *Binance* Testnet;

Versão Final Homologada
27/12/2025 13:45

- USE_TESTNET: Flag para definir o ambiente de testes (True) ou produção (False);
- SYMBOL: Par de trading (padrão: `'BTCUSDT'`);
- TRADE_QUANTITY: Quantidade utilizada em cada operação;
- DEFAULT_MA_FAST: Período da média móvel rápida;
- DEFAULT_MA_SLOW: Período da média móvel lenta;
- DEFAULT_START_DATE_STR: Data inicial para busca de dados históricos.

O sistema foi projetado com uma arquitetura *multithread*, permitindo a execução simultânea de tarefas assíncronas. Essa abordagem garante que operações críticas, como o processamento de dados de mercado e a atualização da interface gráfica, ocorram de forma independente, evitando bloqueios na execução.

A *thread* principal é responsável pela gerência da interface desenvolvida em *Tkinter*. Nela, são processados os eventos de interação do usuário e atualizados os elementos visuais, como displays de preço, gráficos e registros de *log*.

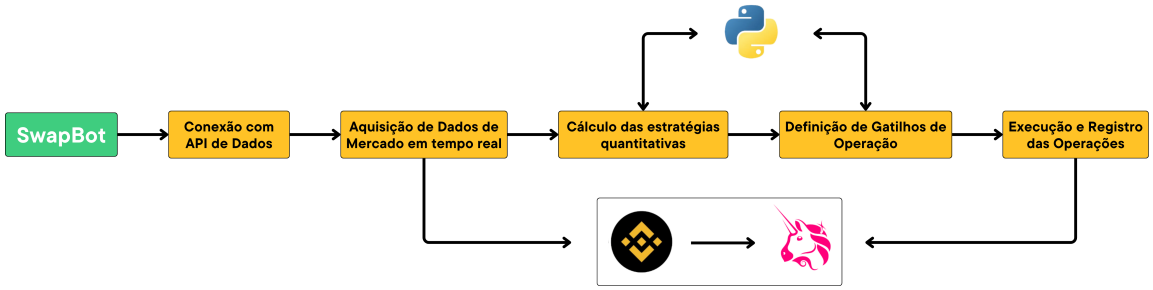
A *thread* de trading executa continuamente a lógica operacional do sistema. Ela realiza a coleta de preços de mercado, o cálculo de indicadores e a execução das ordens de compra e venda.

Paralelamente, a *thread* de monitoramento de preço realiza a coleta periódica dos preços dos ativos e envia essas informações à fila responsável pela atualização dos valores exibidos.

O sistema também conta com uma *thread* de hibernação, que mantém o monitoramento e a atualização de dados sem execução de operações de trading, garantindo estabilidade em períodos de inatividade.

O sistema, portanto, apresenta uma arquitetura estruturada, com separação de responsabilidades entre as camadas de API, lógica e interface.

Versão Final Homologada
27/12/2025 13:45



Fonte: Elaborado pelo autor (2025).

Figura 11 – Arquitetura operacional do sistema de trading automatizado.

O funcionamento da plataforma pode ser descrito como uma conexão via API para aquisição de dados de mercado. Em seguida, os parâmetros relacionados às estratégias são calculados em tempo real e, havendo oportunidade, as operações são executadas automaticamente. Os registros das transações são armazenados em estruturas de dados do tipo dicionário, permitindo rastreabilidade e análise posterior do desempenho das estratégias.

3.5 Livro de Ordens

3.5.1 Descrição

A funcionalidade de Livro de Ordens implementa uma estratégia de negociação baseada em agendamento de operações. O sistema monitora continuamente o preço atual do mercado e executa ordens de compra ou venda quando o preço atinge valores-alvo configurados pelo usuário.

Essa abordagem permite que o operador defina previamente seus pontos de entrada e saída, reduzindo a influência de fatores emocionais nas decisões de investimento.

O Livro de Ordens atua como um intermediário entre o usuário e o mercado, armazenando os parâmetros de cada operação e verificando periodicamente se as condições estabelecidas foram satisfeitas. Quando o preço de um ativo atinge o valor de disparo configurado, o sistema envia automaticamente a instrução de execução.

No contexto de automação financeira, o uso de um sistema baseado em Livro de Ordens permite alcançar diversos resultados que impactam diretamente a eficiência e a confiabilidade das operações.

Versão Final Homologada
27/12/2025 13:45

Entre os principais resultados esperados, destaca-se a execução automática em ambientes *CeFi* e *DeFi*, uma vez que o sistema realiza as operações de compra e venda de forma autônoma, no exato momento em que o preço do ativo atinge os parâmetros previamente definidos pelo usuário.

O sistema é capaz de gerenciar simultaneamente múltiplas ordens, otimizando o tempo de resposta e aumentando o aproveitamento das oportunidades.

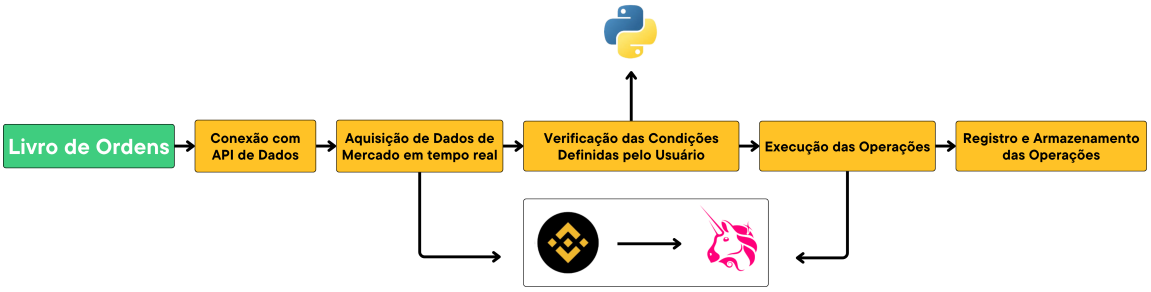
A redução da influência emocional também é um resultado esperado, pois, ao automatizar as operações, o usuário aplica sua estratégia de entradas e saídas sem depender de decisões tomadas em momentos de alta volatilidade.

Além disso, observa-se um aumento da transparência e da rastreabilidade das operações. As ordens são registradas quando executadas e devidamente documentadas nos logs do sistema, permitindo auditorias, análises de desempenho e ajustes de parâmetros com base em dados históricos.

Por fim, destaca-se a integração com estratégias quantitativas, visto que o Livro de Ordens constitui uma base para o desenvolvimento de algoritmos de execução e implementação de estratégias quantitativas.

3.5.2 Aspectos Técnicos do Desenvolvimento

Muitos dos aspectos técnicos associados a execução do Livro de Ordens são os mesmo apresentados no *SwapBot*, uma vez que as duas operações dividem a mesma arquitetura de execução.



Fonte: Elaborado pelo autor (2025).

Figura 12 – Arquitetura operacional do módulo de Livro de Ordens.

O funcionamento do Livro de Ordens pode ser descrito da seguinte forma: inicialmente, estabelece-se uma conexão via API para realizar o monitoramento dos dados. Em seguida, ocorre a verificação das condições

Versão Final Homologada
27/12/2025 13:45

definidas pelo usuário e, quando satisfeitas, o sistema executa automaticamente a ordem.

Versão Final Homologada
27/12/2025 13:45

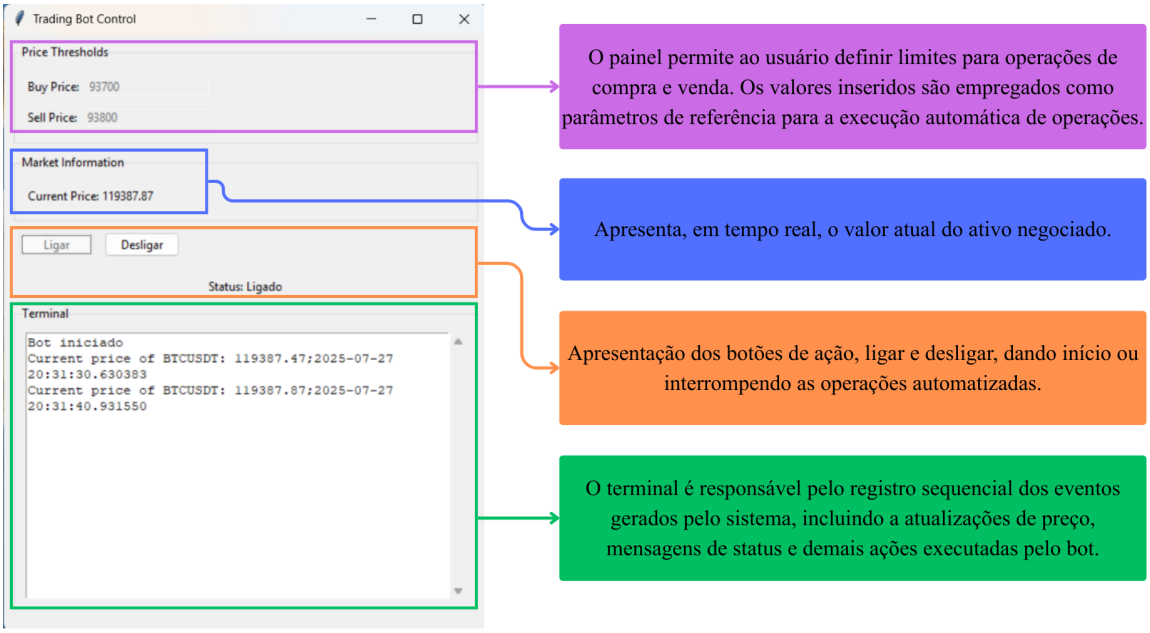
4 Resultados e Discussão

Este capítulo apresenta os resultados obtidos com o desenvolvimento e execução do sistema proposto, bem como uma discussão crítica sobre o desempenho do sistema, a acurácia dos indicadores calculados e a eficácia da interface desenvolvida.

4.1 Interface e Experiência de Usuário

4.1.1 Interface de Operação

Inicialmente, como apresentado na Figura 13, a interface contemplava apenas dois parâmetros configurados pelo usuário, com o intuito de testar as funcionalidades de coleta e armazenamento de dados, bem como a interação do sistema com os mecanismos de execução das operações.



Fonte: Elaborado pelo autor (2025).

Figura 13 – Primeira versão da interface de operação.

Na parte superior da interface, o painel permite ao usuário definir valores para realizar a compra ou venda

Versão Final Homologada
27/12/2025 13:45

do ativo quando o preço escolhido for atingido pelo mercado. Em sequência, há um mostrador responsável por apresentar o preço do criptoativo selecionado para a operação (em dólares). Logo abaixo, encontram-se os botões para ligar e desligar a operação. Por fim, apresenta-se um terminal com os logs da operação, o que possibilita a realização de auditorias e análises dos resultados obtidos pelo mecanismo.

Com o avanço das funcionalidades desenvolvidas, a interface do usuário foi redesenhada para comportar o agendamento de compra e venda, bem como a aplicação da estratégia de cruzamento de médias móveis de forma nativa.



Fonte: Elaborado pelo autor (2025).

Figura 14 – Interface final do sistema.

Como apresentado na Figura 14, a interface final proposta conta com um gráfico do par em operação, com funcionalidades práticas, como zoom e navegação temporal; um painel de configurações que permite ao usuário estabelecer os parâmetros e estratégias quantitativas; botões para iniciar negociações, hibernar (opção responsável por realizar apenas a coleta de dados em tempo real, a fim de verificar o armazenamento correto das informações, a conexão adequada com o meio de operação e a simulação das estratégias em execução em tempo real, mesmo fora de ambientes de redes de teste) e Parar (função que interrompe as operações). Por fim, o terminal que apresenta ao usuário as operações em andamento, assim como possíveis erros na execução

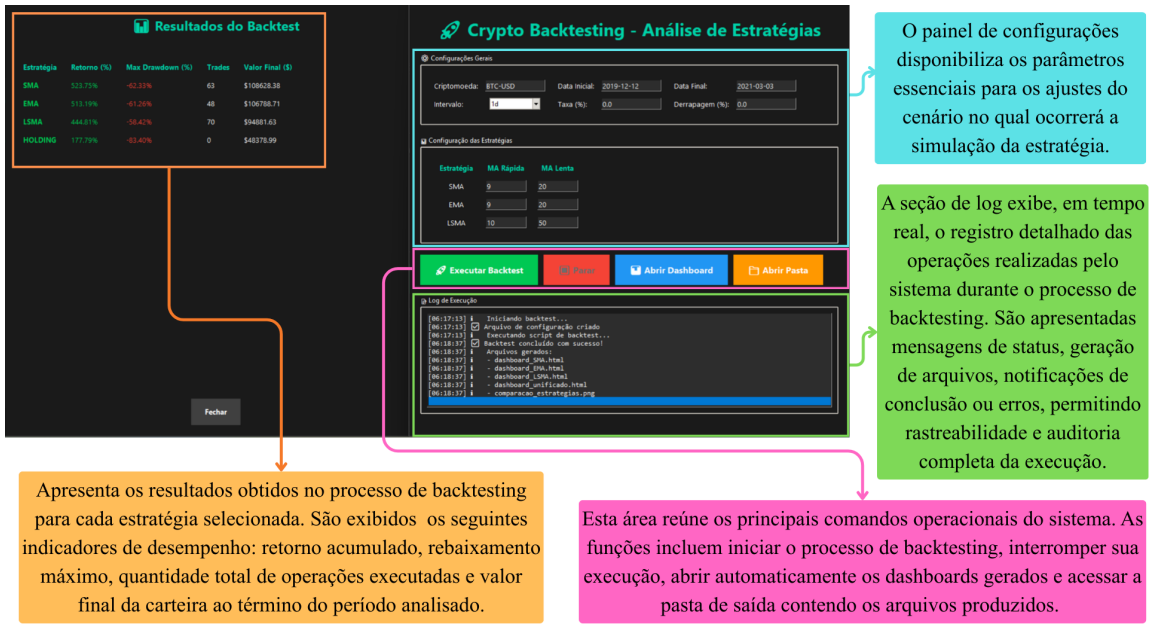
Versão Final Homologada
27/12/2025 13:45

do projeto, foi mantido na versão final.

4.1.2 Interface do Backtest

A construção da interface do *backtest* foi realizada posteriormente à execução dos testes e à coleta dos dados, a fim de permitir, assim como nos mecanismos operacionais, a utilização do sistema por um maior número de usuários.

A Figura 15 apresenta a interface desenvolvida que apresenta aos usuários campos de preenchimento para a execução dos testes com os três tipos de médias móveis simultaneamente.



Fonte: Elaborado pelo autor (2025).

Figura 15 – Interface do módulo de backtest.

O programa apresenta quatro botões em sua interface: um responsável por iniciar os testes, outro destinado a interromper a execução caso o usuário deseje parar um teste em andamento, um botão para abrir a *dashboard* em formato *HTML* gerada ao final da execução e outro para abrir o diretório onde os arquivos referentes ao teste foram salvos.

Dentre os parâmetros que o usuário pode definir, destacam-se: o par de criptoativos selecionado, a data

Versão Final Homologada
27/12/2025 13:45

inicial e final dos testes, o intervalo de coleta dos dados, as taxas de rede ou de operação, a derrapagem (*slippage*) e os períodos das médias móveis desejadas.

Ainda na interface do backtest, após a conclusão dos testes, a aba “Resultados do *Backtest*” é apresentada como uma síntese das análises realizadas.

Os dados coletados durante os testes são exibidos no terminal e armazenados em diretórios próprios após a conclusão. Em seguida, é gerada uma *dashboard* para comparação de desempenho entre as médias móveis e a estratégia de *holding*.

Essa *dashboard* consolida os principais indicadores de desempenho das estratégias avaliadas, permitindo a visualização comparativa de métricas como retorno acumulado, volatilidade, *drawdown* máximo, taxa de acerto e relação risco–retorno. A apresentação gráfica dos resultados facilita a identificação de padrões de comportamento ao longo do período analisado, bem como a avaliação da robustez de cada abordagem frente às variações do mercado.

Além disso, a interface possibilita a análise individual de cada estratégia, oferecendo detalhes sobre as operações executadas, pontos de entrada e saída, bem como o impacto de diferentes configurações de médias móveis nos resultados obtidos. Dessa forma, o ambiente de *backtest* não apenas valida a eficácia das estratégias propostas, mas também fornece subsídios para ajustes finos e otimizações futuras, contribuindo para a tomada de decisão fundamentada no desenvolvimento e aprimoramento do sistema.

4.1.3 Dashboard para comparação de performance entre médias móveis

Com o intuito de integrar todos os dados obtidos nos testes realizados, foi desenvolvida uma *dashboard* interativa para comparação de performance entre as médias móveis e a estratégia de *holding*.

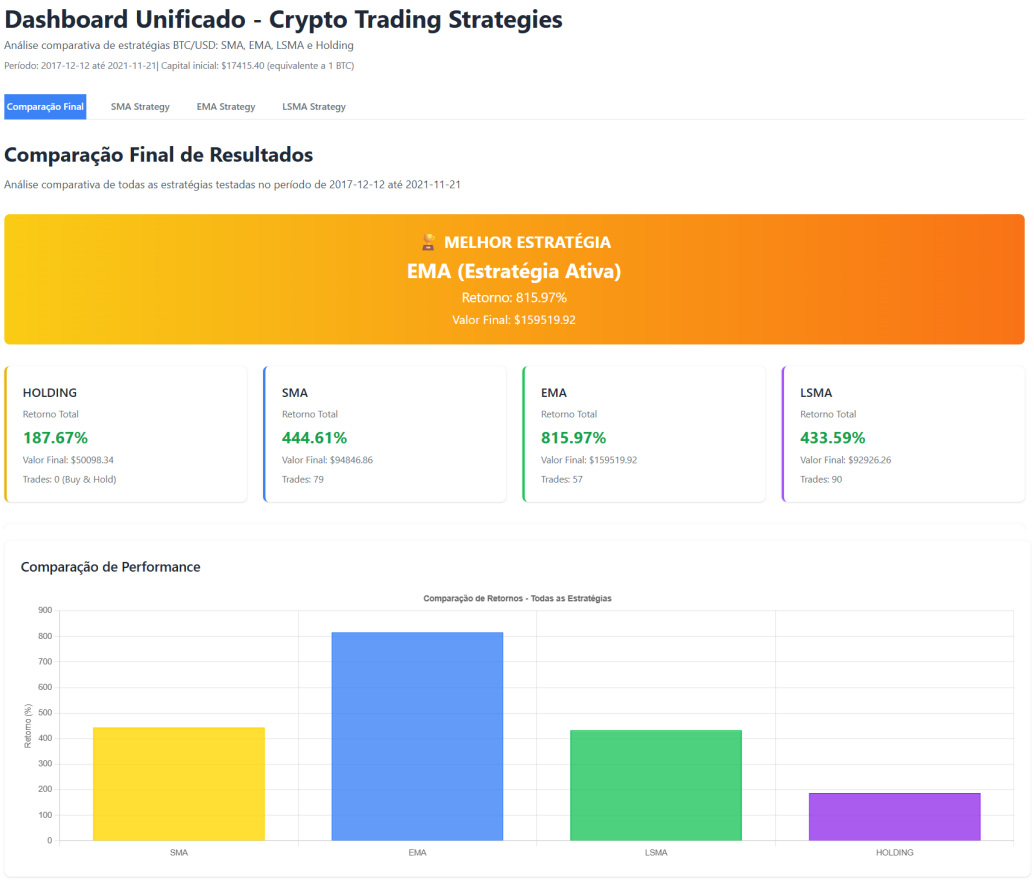
A *dashboard* é apresentada na Figura 16. O cabeçalho da *dashboard* apresenta as informações referentes às datas de início e término da simulação, bem como o preço de uma unidade de *Bitcoin* na data de início dos testes. Esse valor é utilizado comparativamente na análise para identificar o desempenho em relação à quantidade de *tokens* e ao crescimento do patrimônio denominado em dólar.

Logo abaixo, encontram-se as abas que individualmente armazenam os dados de performance de cada média móvel, além da página inicial destinada à comparação entre as estratégias.

Em seguida, é exibido um *banner* horizontal destacando a estratégia que apresentou o melhor desempenho.

Versão Final Homologada
27/12/2025 13:45

Na sequência, são apresentados dados de performance de cada estratégia, tais como rendimento, valor final e total de operações. Esses resultados são comparados visualmente por meio de um histograma.



Fonte: Elaborado pelo autor (2025).

Figura 16 – Página inicial da dashboard.

A Figura 17 apresenta as informações contidas na aba referente ao desempenho da média móvel exponencial (EMA). Inicialmente, observa-se um gráfico em que o traçado preto representa o comportamento do preço do *Bitcoin*, enquanto as médias móveis rápida e lenta são representadas, respectivamente, pelas cores azul e vermelha. Os triângulos indicam os sinais de compra e venda gerados pela estratégia.

Na sequência, apresenta-se um gráfico do comportamento do portfólio em relação ao preço do *Bitcoin*. Nesse gráfico, é possível identificar o comportamento típico de compra, quando o portfólio acompanha as

Versão Final Homologada
27/12/2025 13:45

flutuações do preço, e de venda, observado nos períodos em que o valor do portfólio permanece constante.

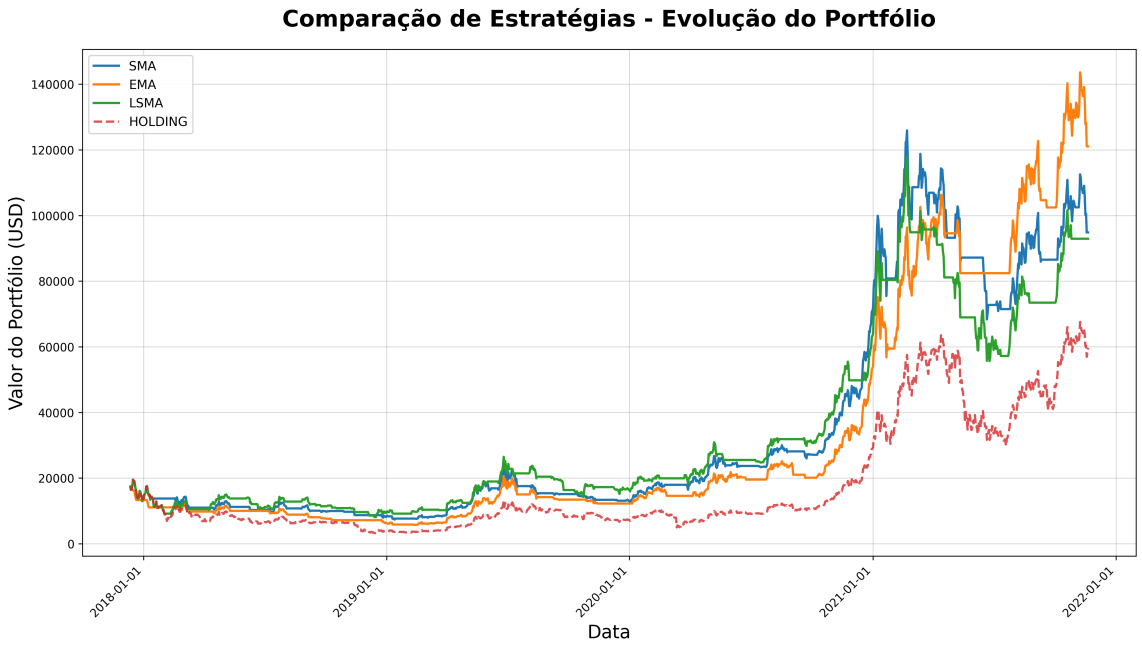


Fonte: Elaborado pelo autor (2025).

Figura 17 – Página da EMA na *dashboard*.

Para verificar o comportamento das médias móveis testadas, apresenta-se também um gráfico de comparação de performance ao longo do tempo, ilustrado na Figura 18. Esse tipo de visualização permite observar como o comportamento intrínseco ao cálculo de cada média móvel pode, ao longo do tempo, gerar resultados positivos ou negativos.

Versão Final Homologada
27/12/2025 13:45



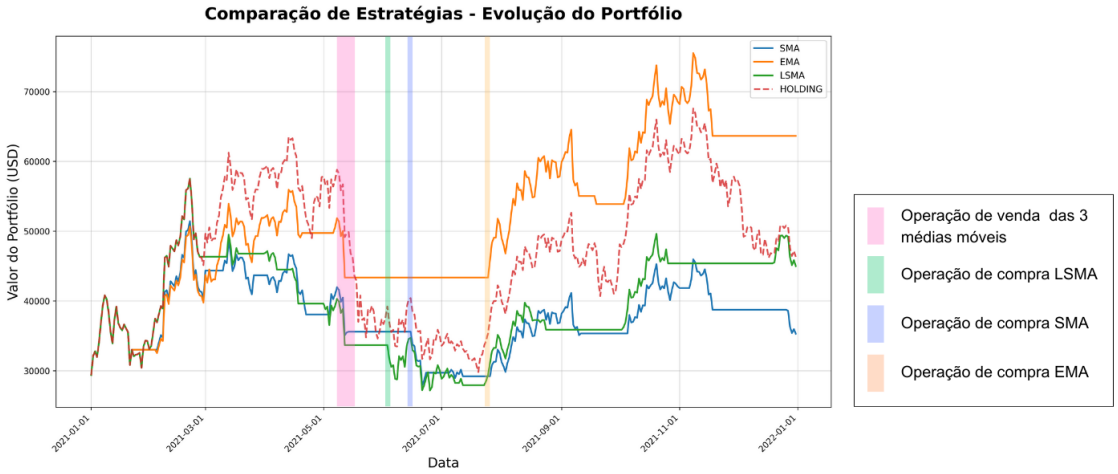
Fonte: Elaborado pelo autor (2025).

Figura 18 – Comparação temporal das estratégias baseadas em médias móveis de 2018 a 2022.

A partir da comparação da evolução temporal das estratégias, é possível observar que, até próximo do início de 2021, a diferença de performance entre as médias móveis não é tão discrepante quanto a partir desse período.

A Figura 19 apresenta o mesmo conjunto de dados da figura anterior, porém com o intervalo reduzido para o período de 01/01/2021 a 01/01/2022. Dessa forma, torna-se possível observar com maior clareza as diferenças na tomada de decisão conforme o tipo de média utilizada. Nesse período, verifica-se uma convergência na operação de venda das três estratégias, seguida de momentos em que cada uma delas executa a operação de compra, conforme o cruzamento de suas médias.

Versão Final Homologada
27/12/2025 13:45



Fonte: Elaborado pelo autor (2025).

Figura 19 – Comparação temporal das estratégias baseadas em médias móveis de 2021 a 2022.

Quanto menor o preço no ato da recompra, maior o acúmulo de *tokens*, o que, em caso de posterior valorização do ativo, representa maior crescimento do portfólio. A ordem das reentradas observadas foi: LSMA, SMA e EMA. O acúmulo percentual de *tokens* na recompra foi, respectivamente, de 22,61%, 17,01% e 30,42%.

Observa-se, ainda, que mesmo em um cenário de tendência de baixa, as estratégias LSMA e SMA apresentaram cruzamentos em picos locais, o que, por consequência, resultou em menor acúmulo de *tokens*.

Dada a recuperação do preço do ativo, a estratégia baseada na EMA, por ter acumulado uma maior quantidade de *tokens*, apresentou o maior valor de portfólio ao final deste período de negociações.

4.2 Análise Quantitativa

Os dados obtidos foram organizados na Tabela 1, distribuídos por período de análise. Essa tabela relaciona a quantidade de operações (*trades*), o rebaixamento máximo (*Max Drawdown*), o crescimento do portfólio em dólar e o crescimento na quantidade de *tokens*.

Versão Final Homologada
27/12/2025 13:45

Tabela 1 – Desempenho dos indicadores técnicos nos diferentes períodos analisados.

Período	Indicador	Total de Trades	Max Drawdown	Crescimento em dólar	Crescimento de Tokens
FF1	SMA	70	61,98%	+2235,99%	+28,31%
	EMA	62	66,91%	+1604,56%	-6,37%
	LSMA	73	45,94%	+2857,77%	+62,46%
FF2	SMA	74	62,88%	+565,75%	+32,27%
	EMA	48	57,30%	+1051,49%	+128,77%
	LSMA	85	55,79%	+1907,50%	+20,69%
FT1	SMA	47	32,65%	+6043,61%	-43,88%
	EMA	43	34,98%	+5050,74%	-52,95%
	LSMA	45	28,71%	+4975,43%	-53,64%
FT2	SMA	55	46,39%	+1488,55%	-23,22%
	EMA	33	40,60%	+2572,64%	+29,17%
	LSMA	63	52,77%	+968,64%	-48,35%
FT3	SMA	51	31,39%	+270,04%	-50,26%
	EMA	39	24,95%	+385,02%	-34,80%
	LSMA	57	30,18%	+242,02%	-54,02%
TF1	SMA	23	54,50%	-54,50%	+168,59%
	EMA	17	64,29%	-64,29%	+110,77%
	LSMA	26	63,66%	-62,38%	+122,07%
TF2	SMA	19	61,74%	-61,47%	+53,72%
	EMA	15	67,17%	-66,93%	+31,92%
	LSMA	22	60,01%	-59,17%	+62,59%
TT1	SMA	78	61,63%	+482,88%	+66,57%
	EMA	54	71,47%	+608,64%	+102,53%
	LSMA	89	63,66%	+330,46%	+23,01%
TT2	SMA	70	63,81%	+32,67%	-26,58%
	EMA	54	18,48%	+47,30%	-18,48%
	LSMA	81	60,01%	+32,19%	-26,84%

Fonte: Elaborado pelo autor (2025).

Os períodos Fundo–Fundo, Fundo–Topo, Topo–Fundo e Topo–Topo foram representados por FF, FT, TF e TT, respectivamente. Dessa forma:

- FF1: 14/01/2015 a 15/12/2018
- FF2: 15/12/2018 a 21/11/2022
- FT1: 14/01/2015 a 17/12/2017
- FT2: 15/12/2018 a 10/11/2021
- FT3: 21/11/2022 a 22/07/2025

Versão Final Homologada
27/12/2025 13:45

- TF1: 17/12/2017 a 15/12/2018
- TF2: 10/11/2021 a 21/11/2022
- TT1: 17/12/2017 a 10/11/2021
- TT2: 10/11/2021 a 22/07/2025

Considerando o total de operações realizadas, é possível notar que, em quase todos os testes, a média móvel calculada por mínimos quadrados com ajuste linear se apresentou como a mais ativa. Por outro lado, o uso da média móvel com ajuste exponencial apresentou, em todos os casos, uma menor quantidade de operações, demonstrando menor sensibilidade a pequenas flutuações de preço.

O *Max Drawdown* é uma métrica utilizada para mensurar a maior perda percentual de um pico a um fundo do portfólio. Nesse sentido, é usual considerar que quanto menor o *Max Drawdown*, menos arriscada é a estratégia testada. Contudo, é possível observar que os períodos FF e TF apresentam valores elevados. Isso ocorre porque, para ambos, o *Max Drawdown* é calculado entre o fundo mais recente e a alta histórica anterior, não necessariamente indicando maior risco na operação, mas ressaltando que a percepção do usuário sobre o momento do ciclo em que o mercado se encontra deve orientar a decisão pela utilização, ou não, da estratégia de cruzamento de médias móveis.

O crescimento em dólar mensura o valor do portfólio em moeda fiduciária ao final dos testes, independentemente da posição atual no par. Dessa forma, é possível verificar o crescimento nominal do investimento no período analisado.

O crescimento de tokens refere-se à quantidade final de tokens em relação à quantidade inicial (1 BTC). Essa métrica permite avaliar o desempenho da estratégia sem considerar oscilações de curto prazo no preço do ativo. Caso o usuário opte por carregar seus ativos no longo prazo, essa métrica representa diretamente o rendimento percentual associado à posse do ativo.

4.3 Análise Qualitativa

Para além de uma análise quantitativa, também podemos destacar alguns aspectos qualitativos apresentados pelos diferentes métodos de cálculo das médias móveis. Na Tabela 2, são apresentadas algumas características e como elas se associam a cada tipo de média. Entre esses aspectos estão o cálculo, a sensibilidade ao preço, o atraso na percepção de movimentos de mercado, o nível de suavização da série financeira, o cenário ideal de operação, além das principais vantagens e desvantagens.

Versão Final Homologada
27/12/2025 13:45

Tabela 2 – Diferenças qualitativas entre as médias móveis

Característica	SMA	EMA	LSMA
Cálculo	Média aritmética simples dos preços em um período definido.	Média ponderada que atribui maior peso aos preços mais recentes.	Linha de regressão linear calculada sobre os preços do período.
Sensibilidade ao Preço	Baixa. Reage lentamente a novas informações, pois todos os pontos de dados têm o mesmo peso.	Alta. Reage rapidamente a mudanças recentes de preço devido à ponderação exponencial.	Muito alta. Projeta a tendência futura com base nos dados passados, tornando-a preditiva e bastante sensível.
Atraso	Alto. É o indicador com maior atraso em relação ao preço atual.	Moderado. Reduz o atraso em comparação com a SMA, acompanhando o preço mais de perto.	Baixo. Frequentemente se move à frente do preço, pois não é uma média, mas uma projeção de tendência.
Suavização	Alta. Produz a linha mais suave, filtrando bem o ruído do mercado.	Moderada. É mais suave que o preço, mas menos suave que a SMA.	Baixa. Pode ser mais "nervosa" ou volátil, pois segue a inclinação da tendência imediata.
Ideal Para	Mercados com tendências claras e duradouras, nos quais o ruído precisa ser filtrado.	Mercados que exigem resposta mais rápida, como operações de curto prazo ou cenários de alta volatilidade.	Identificar a força e a direção imediata da tendência. Útil para captar o *momentum* do mercado.
Principal Vantagem	Simplicidade e eficácia na remoção de ruído.	Equilíbrio entre suavização e capacidade de resposta rápida.	Capacidade de antecipar movimentos e reduzir o atraso.
Principal Desvantagem	Atraso significativo, que pode gerar sinais de entrada e saída tardios.	Pode gerar sinais falsos em mercados voláteis ou sem tendência definida.	Por ser preditiva, pode ultrapassar o preço e produzir sinais prematuros ou falsos em mercados laterais.

Fonte: Elaborado pelo autor (2025).

4.4 Limitações e perspectivas futuras

Durante o desenvolvimento e a execução dos testes, foram identificadas algumas limitações e fragilidades no sistema proposto. Para que a metodologia apresentada possa ser aplicada em um ambiente real de automação financeira, torna-se necessário o aprimoramento de determinados módulos e funcionalidades dos programas desenvolvidos.

Versão Final Homologada
27/12/2025 13:45

Uma preocupação essencial diz respeito à segurança operacional. Considerando que este trabalho tem como objetivo investigar e apresentar um modelo conceitual para automação financeira de criptoativos, a implementação de tecnologias voltadas à proteção e à segurança das operações não foi contemplada no projeto.

Para a migração dos mecanismos operacionais, faz-se necessária a alteração da forma como o *software* se comunica com a rede, substituindo as credenciais de acesso utilizadas na rede de testes por uma infraestrutura compatível com uma *Blockchain* real.

A estratégia de cruzamento de médias móveis foi empregada como mecanismo para demonstrar as funcionalidades do sistema proposto, bem como sua integração com módulos de tratamento e análise de dados. Dessa forma, embora o sistema possua integração nativa com estratégias baseadas em médias móveis, não é objetivo deste trabalho emitir juízo, no âmbito das ciências econômicas, acerca da efetividade dessa aplicação. O foco está no desenvolvimento de meios que permitam a implementação automatizada de estratégias quantitativas, em geral, em diferentes mercados, especialmente no mercado *DeFi*.

Versão Final Homologada
27/12/2025 13:45

5 Conclusão

O presente trabalho dedicou-se à realização de uma revisão bibliográfica sobre criptomoedas e as oportunidades do mercado de finanças descentralizadas, bem como ao desenvolvimento e à validação de um sistema automatizado para a aplicação de estratégias quantitativas no mercado de criptoativos. Para a negociação, utilizou-se o *Bitcoin* (BTC), o ativo com maior capitalização de mercado.

A investigação demonstrou a viabilidade e a eficácia da utilização de mecanismos de automação, como o SwapBot, para executar operações de compra e venda baseadas em indicadores técnicos, especificamente o cruzamento de médias móveis. A metodologia proposta englobou o desenvolvimento das ferramentas de Backtest, Livro de Ordens e SwapBot. Essa estrutura mostrou-se robusta para simular, agendar e executar estratégias de forma autônoma e síncrona, tanto em mercados tradicionais quanto em mercados de criptoativos operando em redes descentralizadas.

A análise comparativa entre diferentes tipos de médias móveis (simples, exponencial, e de mínimos quadrados com aproximação linear) permitiu identificar oportunidades associadas a cada abordagem em distintos momentos dos ciclos de mercado. Os resultados obtidos nas simulações de backtesting indicaram que, embora nenhuma estratégia apresente resultados positivos em todos os cenários, períodos caracterizados por queda seguida de recuperação tendem a gerar maior acúmulo de tokens e, consequentemente, maior valor de portfólio final. Conclui-se, portanto, que a escolha do tipo de média e dos períodos de cálculo é um fator determinante para o desempenho do sistema; dessa forma, a utilização de mecanismos que otimizem esses parâmetros mostra-se útil para aumentar a eficiência do processo de tomada de decisão.

A arquitetura do sistema foi desenvolvida em Python, utilizando bibliotecas de código aberto. Essa arquitetura mostrou-se flexível e eficiente, permitindo a integração com a rede de testes (*Binance Testnet*), a modificação do módulo de comunicação para interação com protocolos *DeFi* e a construção de interfaces gráficas (GUI) para configuração de parâmetros, execução das operações e análise dos dados obtidos. De forma geral, considera-se que o desenvolvimento de interfaces acessíveis a usuários sem conhecimento aprofundado em programação foi realizado com êxito, contribuindo para a democratização do acesso a tecnologias de automação financeira.

Como trabalhos futuros, sugere-se a expansão do sistema para operar nativamente em ambientes *DeFI*, explorando a interação direta com contratos inteligentes em redes *Blockchain* compatíveis com corretoras

Versão Final Homologada
27/12/2025 13:45

descentralizadas (DEX) e outros aplicativos descentralizados (DApps). Adicionalmente, propõe-se a implementação de um sistema de otimização de parâmetros, como os períodos das médias rápida e lenta, utilizando mecanismos de *machine learning*, possibilitando simulações mais abrangentes e aderentes a diferentes cenários de mercado.

Considerando a hospedagem do *software* desenvolvido, recomenda-se ainda a elaboração de um h*ardware open-source* de baixo custo e baixo consumo energético, visando à autocustódia completa do sistema. Essa solução garantiria maior autonomia ao usuário, eliminando a necessidade de hospedagem em nuvem, reduzindo o custo operacional e aumentando a segurança das operações.

Por fim, conclui-se que o trabalho atingiu seus objetivos ao demonstrar a viabilidade técnica e prática de um sistema de automação voltado à negociação quantitativa em mercados de criptoativos. Os resultados apresentados evidenciam como a integração de conceitos da Engenharia Física, como modelagem matemática, análise de sinais, simulação computacional, métodos numéricos e tratamento de dados. A utilização de métodos quantitativos, aliada à capacidade de projetar, validar e otimizar sistemas complexos, reforça o papel multidisciplinar da Engenharia Física, abrindo espaço para pesquisas futuras e para o aprimoramento de tecnologias capazes de fortalecer o ecossistema *DeFi* e ampliar o acesso a ferramentas de automação financeira.

Versão Final Homologada
27/12/2025 13:45

Referências

AL-AMRI, R. et al. Cryptocurrency adoption: current stage, opportunities, and open challenges. **International Journal of Advanced Computer Research**, v. 9, n. 44, p. 293–307, 2019.

ALAMSYAH, A.; KUSUMA, G. N. W.; RAMADHANI, D. P. A review on decentralized finance ecosystems. **Future Internet**, v. 16, n. 3, p. 76, 1 mar. 2024.

ANTONOPOULOS, Andreas M. *Mastering Bitcoin: Unlocking Digital Crypto-Currencies*. Sebastopol, CA: O'Reilly Media, 2015.

ANTONOPOULOS, Andreas M. *The Internet of Money: A Collection of Talks*. Createspace Independent Publishing Platform, 2016.

BACK, Adam. *Hashcash — A Denial of Service Counter-Measure*. Cypherspace, 2002. Disponível em: <http://www.hashcash.org/hashcash.pdf>. Acesso em: 25 nov. 2025.

BADEV, Anton I. Interconnected DeFi: Ripple effects from the Terra collapse. 2022. Disponível em: <https://www.federalreserve.gov/econres/feds/files/2023044pap.pdf>. Acesso em: 29 nov. 2025.

BAKER, M.; WURGLER, J. Investor sentiment in the stock market. **Journal of Economic Perspectives**, v. 21, n. 2, p. 129–151, 2007.

BHUTORIA, R.; McCURDY, T.; WARHOLAKADDING, M. **Bitcoin to a Corporate Treasury**. Fidelity Digital Assets, [2025]. Disponível em: <https://www.fidelitydigitalassets.com/research-and-insights/adding-bitcoin-corporate-treasury>. Acesso em: 29 set. 2025.

BITCOIN CORE. *Statement*. 7 jan. 2016. Disponível em: <https://bitcoincore.org/en/2016/01/07/statement/>. Acesso em: 19 nov. 2025.

BITCOIN MAGAZINE. *How Did Satoshi Think of Bitcoin?* Autor: Unchained Capital. 13 abr. 2024. Disponível em: <https://bitcoinmagazine.com/technical/how-did-satoshi-think-of-bitcoin>. Acesso em: 25 nov. 2025.

BLOCKCHAIN Technology: Beyond Bitcoin. **Satoshi Speaks**, [2024]. Disponível em: <https://satoshispeaks.com/blockchain-technology-beyond-bitcoin/>. Acesso em: 25 nov. 2025.

BOURI, E. et al. On the hedge and safe haven properties of Bitcoin: Is it really more than a diversifier? **Finance Research Letters**, v. 20, n. 20, p. 192–198, fev. 2017.

BROOKS, J. C. **Mastering Technical Analysis**: Using the Tools of Technical Analysis for Profitable Trading. New York: McGraw-Hill, 2006.

Versão Final Homologada
27/12/2025 13:45

BTCGPU PROJECT. *Bitcoin Gold: Secure, Democratized, Decentralized*. 2017. Disponível em: <https://resources.cryptocompare.com/asset-management/137/1692716432609.pdf>. Acesso em: 19 nov. 2025.

CALZA, A. S. *A atividade dos mineradores de Bitcoin frente os diferentes cenários: uma análise microeconômica*. 2023. Trabalho de Conclusão de Curso (Graduação em Ciências Econômicas) — Universidade Federal de Santa Catarina, Florianópolis, 2023.

CARVALHO, C. E.; PIRES, D. A.; ARTIOLI, M.; OLIVEIRA, G. C. Cryptocurrencies: technology, initiatives of banks and central banks, and regulatory challenges. **Economia e Sociedade**, v. 30, n. 2, p. 467–496, 2021. Disponível em: <https://doi.org/10.1590/1982-3533.2021v30n2art08>. Acesso em: 19 nov. 2025.

CHABOUD, A. P. et al. Rise of the machines: Algorithmic trading in the foreign exchange market. **The Journal of Finance**, v. 69, n. 5, p. 2045–2084, 2014.

DAI, Wei. *B-Money*. 1998. Disponível em: <https://cdn.nakamotoinstitute.org/docs/b-money.txt>. Acesso em: 29 set. 2025.

ETORO. *Fear vs Optimism: The Top Emotions Driving Retail Investors' Decisions*. London: eToro, 8 maio 2025. Disponível em: <https://www.etoro.com/news-and-analysis/etoro-updates/fear-vs-optimism-the-top-emotions-driving-retail-investors-decisions/>. Acesso em: 25 nov. 2025.

GOGOL, K. et al. Empirical and theoretical analysis of liquid staking protocols. 2024. Disponível em: <https://arxiv.org/abs/2401.16353>. Acesso em: 29 nov. 2025.

GORTON, G. B. *Slapped in the Face by the Invisible Hand: Banking and the Panic of 2007*. New Haven: Yale University, 2010.

HARVEY, C. R. Cryptocurrencies / Innovation and Cryptoventures. Duke University & NBER, 21 jan. 2018. Disponível em: <https://people.duke.edu/~charvey/Teaching/897_2018/2_Cryptocurrency/Cryptocurrencies_January_21_2018.pdf>. Acesso em: 25 nov. 2025.

HAYES, A. S. Bitcoin price and its marginal cost of production: Support for a fundamental value. **Applied Economics Letters**, v. 26, n. 7, p. 554–560, 2019.

HO, A.; RUAN, M. Tokenized real-world assets – Part 1: Landscape & value chain. Animoca Brands, 2025.

JPMORGAN targeting a Q4 rollout for its AI equities utility, LOXM. **Finance Magnates**, 1 ago. 2017. Disponível em: <https://www.financemagnates.com/institutional-forex/execution/jpmorgan-targeting-q4-rollout-ai-equities-utility-loxm/>. Acesso em: 25 nov. 2025.

KAHNEMAN, D.; TVERSKY, A. Prospect theory: An analysis of decision under risk. **Econometrica**, v. 47, n. 2, p. 263–291, 1979. DOI: 10.2307/1914185.

Versão Final Homologada
27/12/2025 13:45

LEHNER, E.; HUNZEKER, D.; ZIEGLER, J. R. Funding science with science: Cryptocurrency and independent academic research funding. **Ledger**, v. 2, p. 65–76, 2017. DOI: 10.5915/LEDGER.2017.108.

MANKIW, N. G. *Introdução à Economia*. 2009.

MISHKIN, F. S. *The Economics of Money, Banking, and Financial Markets*. 13. ed. Pearson Education, 2002, p. 97–104.

MITA, M. et al. What is stablecoin?: A survey on its mechanism and potential as decentralized payment systems. Disponível em: <https://arxiv.org/abs/1906.06037v2>. Acesso em: 29 nov. 2025.

NAKAMOTO, S. *Bitcoin: Um sistema de dinheiro eletrônico peer-to-peer*. 2008. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Acesso em: 25 nov. 2025.

NARAYANAN, A. et al. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton: Princeton University Press, 2016.

NOTLAND, J.; NOWOSTAWSKI, M.; LI, J. Runtime evolution of Bitcoin's consensus rules. **IEEE Transactions on Software Engineering**, v. 49, n. 9, p. 4477–4495, 2023. DOI: 10.1109/TSE.2023.3304851.

OUTUMURO, L. *A Comprehensive Overview of DeFi in 2020*. 2020. Disponível em: <https://app.intotheblock.com/static-reports/bitstamp/defi/2020-overview/report/a-comprehensive-overview-of-defi-in-2020.pdf>. Acesso em: 29 nov. 2025.

POLIZU, C. et al. Are crypto markets correlated with macroeconomic factors? S.l.: S&P Global, 2023. Disponível em: <https://www.spglobal.com/en/research-insights/special-reports/are-crypto-markets-correlated-with-macroeconomic-factors>. Acesso em: 29 nov. 2025.

QIN, K. et al. CeFi vs. DeFi — Comparing centralized to decentralized finance. arXiv:2106.08157, 16 jun. 2021.

QIU, J.; WANG, B.; ZHOU, C. Forecasting stock prices with long-short term memory neural network based on attention mechanism. **PLOS One**, v. 15, n. 1, e0227222, 2020. Disponível em: <https://doi.org/10.1371/journal.pone.0227222>.

RAMIRO, A.; DE QUEIROZ, R. J. G. B. Cypherpunk. **Internet Policy Review**, v. 11, n. 2, p. 1–17, 2022. DOI: 10.14763/2022.2.1664.

REIFF, N. The collapse of FTX: What went wrong with the crypto exchange? Investopedia, 2023. Disponível em: <https://www.investopedia.com/what-went-wrong-with-ftx-6828447>. Acesso em: 29 nov. 2025.

RESENDE, V. *Smart Contracts: O que são e como funcionam?* **Jusbrasil**, [2025]. Disponível em: <https://www.jusbrasil.com.br/artigos/smart-contracts-o-que-sao-e-como-funcionam/1290434477>. Acesso em: 28 out. 2025.

Versão Final Homologada
27/12/2025 13:45

RIPPLE. *BCG Tokenization Report*. [2024]. Disponível em: <https://ripple.com/lp/bcg-tokenization-report/>. Acesso em: 25 nov. 2025.

SATOSHI SPEAKS. *Early Digital Cash Predecessors*. Autor: Nicolas van Saberhagen. [2024]. Disponível em: <https://satoshispeaks.com/early-digital-cash-predecessors/>. Acesso em: 25 nov. 2025.

SCHÄR, F. Decentralized finance: On blockchain- and smart contract-based financial markets. **Federal Reserve Bank of St. Louis Review**, v. 103, n. 2, p. 153–174, 2021.

SEIDEL, M. D. L. Questioning centralized organizations in a time of distributed trust. **Journal of Management Inquiry**, v. 27, n. 1, p. 40–44, 2018.

SHARMA, G. D. et al. Emergence of Bitcoin as an investment alternative: A systematic review and research agenda. **International Journal of Business and Information**, v. 14, n. 1, p. 47–84, 2019. DOI: 10.6702/ijbi.201903$_1$4(1).0003.

SHEFRIN, H. *Beyond Greed and Fear: Understanding Behavioral Finance and the Psychology of Investing*. New York: Oxford University Press, 2007.

SPITZNAGEL, M. *The Dao of Capital: Austrian Investing in a Distorted World*. Hoboken: Wiley, 2013.

STANFORD Encyclopedia of Bitcoin — SegWit Upgrade. Stanford Blockchain, 2024.

STEVENS, R. The 2020 year in review: DeFi. 2020. Disponível em: <https://decrypt.co/52298/the-2020-year-in-review-defi>. Acesso em: 29 nov. 2025.

SWAN, M. *Blockchain: Blueprint for a New Economy*. Sebastopol, CA: O'Reilly Media, 2015.

SZABO, N. *Smart Contracts: Building Blocks for Digital Markets*. 1996. Disponível em: <https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html>. Acesso em: 28 out. 2025.

SZABO, N. *Bit Gold*. Nakamoto Institute, 29 dez. 2005. Disponível em: <https://nakamotoinstitute.org/library/bit-gold/>. Acesso em: 25 nov. 2025.

SZABO, N. *Bit Gold Markets*. Nakamoto Institute, 2008. Disponível em: <https://nakamotoinstitute.org/library/bit-gold-markets/>. Acesso em: 25 nov. 2025.

TAPSCOTT, D.; TAPSCOTT, A. **Blockchain Revolution**: How the technology behind Bitcoin is changing money, business, and the world. New York: Penguin Random House, 2016.

WEN, H. et al. Stablecoins and the emerging hybrid monetary ecosystems. Disponível em: <https://arxiv.org/abs/2505.10997v3>. Acesso em: 29 nov. 2025.

WU, D. Institutional adoption and correlation dynamics: Bitcoin's evolving role in financial markets. 2025. Disponível em: <https://arxiv.org/abs/2501.09911v1>. Acesso em: 29 nov. 2025.

Versão Final Homologada
27/12/2025 13:45

ZAHARUDIN, K. Z.; YOUNG, M. R.; HSU, W. High-frequency trading: Definition, implications, and controversies. **Journal of Economic Surveys**, v. 36, n. 1, p. 75–107, 2021.

Versão Final Homologada
27/12/2025 13:45

ANEXO A – Código-fonte do Bot de Trading

Este apêndice apresenta o código-fonte desenvolvido pelo autor para a implementação de um bot de trading automatizado para o mercado de criptomoedas, com integração à API da corretora Binance, interface gráfica em *Tkinter* e suporte a estratégias baseadas em livro de ordens e cruzamento de médias móveis.

```
import os
import time
import datetime
import csv
import tkinter as tk
from tkinter import ttk, messagebox, scrolledtext
import threading
import queue
import numpy as np
import pandas as pd
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.backends.backend_tkagg import NavigationToolbar2Tk
from binance.client import Client
from binance.exceptions import BinanceAPIException, BinanceRequestException
import traceback

API_KEY = "..."
SECRET_KEY = "..."


USE_TESTNET = True

SYMBOL = 'BTCUSDT'
TRADE_QUANTITY = 0.001

DEFAULT_MA_FAST = 7
```

Versão Final Homologada
27/12/2025 13:45

```
DEFAULT_MA_SLOW = 25

DEFAULT_START_DATE_STR = "1 Jan, 2020"

client = None

def initialize_binance_client():
    """Inicializa o cliente Binance e verifica a conexão."""
    global client
    if API_KEY == "YOUR_API_KEY_PLACEHOLDER" or SECRET_KEY == "
    YOUR_SECRET_KEY_PLACEHOLDER":
        local_api_key = os.environ.get("BINANCE_API_KEY")
        local_secret_key = os.environ.get("BINANCE_SECRET_KEY")
        if not local_api_key or not local_secret_key:
            messagebox.showerror("Erro de Chave da API",
                                 "Chave/Secret da API não encontrada. \n\n" + \
                                 "Por favor, defina as variáveis de ambiente
    BINANCE_API_KEY e BINANCE_SECRET_KEY.")
            return False
        current_api_key = local_api_key
        current_secret_key = local_secret_key
    else:
        current_api_key = API_KEY
        current_secret_key = SECRET_KEY
        print("Aviso: Usar chaves da API diretamente do script não é
    recomendado. Use variáveis de ambiente.")

    try:
        client = Client(current_api_key, current_secret_key, testnet=
    USE_TESTNET)
        client.get_account()
        print("Cliente Binance inicializado com sucesso.")
        return True
    except (BinanceAPIException, BinanceRequestException) as e:
        messagebox.showerror("Erro de Conexão da API", f"Falha ao conectar com
    a Binance: {e}")
        client = None
        return False
    except Exception as e:
        messagebox.showerror("Erro de Inicialização", f"Ocorreu um erro
```

```
    inesperado: {e}")
        client = None
        return False

def get_current_price(symbol):
    """Busca o preço atual para um símbolo com tratamento de erro."""
    if not client:
        return None
    try:
        ticker = client.get_symbol_ticker(symbol=symbol)
        return float(ticker['price'])
    except (BinanceAPIException, BinanceRequestException) as e:
        print(f"Erro ao buscar preço para {symbol}: {e}")
        return None
    except Exception as e:
        print(f"Erro inesperado ao buscar preço: {e}")
        return None

def get_order_book(symbol, limit=100):
    """Busca o livro de ordens (order book) para um símbolo com tratamento de
    erro.

    Args:
        symbol: Par de trading (ex: 'BTCUSDT')
        limit: Número de ordens a retornar (padrão 100, máximo 5000)

    Returns:
        Dict com 'bids' (compras) e 'asks' (vendas) ou None em caso de erro
        Cada ordem é [preço, quantidade]
    """
    if not client:
        return None
    try:
        order_book = client.get_order_book(symbol=symbol, limit=limit)

        bids = [[float(price), float(qty)] for price, qty in order_book['bids
    ']]
        asks = [[float(price), float(qty)] for price, qty in order_book['asks
    ']]
        return {
```

```
            'bids': bids,
            'asks': asks
            }
    except (BinanceAPIException, BinanceRequestException) as e:
        print(f"Erro ao buscar order book para {symbol}: {e}")
        return None
    except Exception as e:
        print(f"Erro inesperado ao buscar order book: {e}")
        return None

def place_buy_order(symbol, quantity, gui_instance):
    """Coloca uma ordem de compra a mercado com tratamento de erro e logging
    ."""
    if not client:
        gui_instance.log_message("Erro: Cliente Binance não inicializado.")
        return None
    try:
        gui_instance.log_message(f"Tentando colocar ordem de COMPRA para {
    quantity} {symbol}...")
        try:
            order = client.order_market_buy(symbol=symbol, quantity=quantity)
        except BinanceAPIException as api_e:
             if USE_TESTNET and "Market orders are not supported" in str(api_e):

                gui_instance.log_message("order_market_buy falhou no testnet,
    tentando create_order...")
                order = client.create_order(symbol=symbol, side=Client.
    SIDE_BUY, type=Client.ORDER_TYPE_MARKET, quantity=quantity)
             else:
                raise api_e

        gui_instance.log_message(f"Ordem de COMPRA bem-sucedida: {order}")
        return order
    except (BinanceAPIException, BinanceRequestException) as e:
        gui_instance.log_message(f"Ordem de COMPRA falhou: {e}")
        return None
    except Exception as e:
        gui_instance.log_message(f"Erro inesperado ao colocar ordem de COMPRA:
    {e}")
        return None
```

```
def place_sell_order(symbol, quantity, gui_instance):
    """Coloca uma ordem de venda a mercado com tratamento de erro e logging."""
    if not client:
        gui_instance.log_message("Erro: Cliente Binance não inicializado.")
        return None
    try:
        gui_instance.log_message(f"Tentando colocar ordem de VENDA para {
    quantity} {symbol}...")
        try:
            order = client.order_market_sell(symbol=symbol, quantity=quantity)
        except BinanceAPIException as api_e:
            if USE_TESTNET and "Market orders are not supported" in str(api_e):
                gui_instance.log_message("order_market_sell falhou no testnet,
    tentando create_order...")
                order = client.create_order(symbol=symbol, side=Client.
    SIDE_SELL, type=Client.ORDER_TYPE_MARKET, quantity=quantity)
            else:
                raise api_e
        gui_instance.log_message(f"Ordem de VENDA bem-sucedida: {order}")
        return order
    except (BinanceAPIException, BinanceRequestException) as e:
        gui_instance.log_message(f"Ordem de VENDA falhou: {e}")
        return None
    except Exception as e:
        gui_instance.log_message(f"Erro inesperado ao colocar ordem de VENDA: {
    e}")
        return None

def get_all_historical_data(symbol, interval, start_str):
    """Busca todos os dados históricos disponíveis desde start_str usando
    paginação."""
    if not client:
        print("Erro: Cliente Binance não inicializado.")
        return None

    print(f"Buscando todos os dados históricos para {symbol} ({interval}) começ
    ando de {start_str}...")
    limit = 1000
    all_klines = []
```

```
    start_ts = int(datetime.datetime.strptime(start_str, '%d %b, %Y').timestamp
    () * 1000)
    end_ts = int(datetime.datetime.now().timestamp() * 1000)

    while True:
        try:
            print(f"Buscando klines de {datetime.datetime.fromtimestamp(
    start_ts/1000)}...")
            klines = client.get_historical_klines(symbol, interval, start_ts,
    limit=limit)

            if not klines:
                print("Nenhuma kline recebida.")
                break

            all_klines.extend(klines)
            last_kline_ts = klines[-1][0]

            if last_kline_ts == start_ts:
                 print("Aviso: Timestamp da última kline corresponde ao tempo
    de início da requisição. Possível lacuna de dados ou fim atingido.")

                 interval_ms = interval_to_milliseconds(interval)
                 if interval_ms:
                     start_ts = last_kline_ts + interval_ms
                 else:
                     print("Não foi possível determinar a duração do intervalo,
     parando busca.")
                     break
            else:

                start_ts = last_kline_ts + 1

            print(f"Buscadas {len(klines)} klines. Total: {len(all_klines)}. Pr
    óximo início: {datetime.datetime.fromtimestamp(start_ts/1000)}")


            time.sleep(0.2)

        except (BinanceAPIException, BinanceRequestException) as e:
```

```
            print(f"Erro da API ao buscar dados históricos: {e}")

            if "Rate limit exceeded" in str(e):
                print("Limite de taxa atingido, aguardando 60 segundos...")
                time.sleep(60)
                continue
            else:
                messagebox.showerror("Erro da API", f"Falha ao buscar dados
    históricos: {e}")
                return None
        except Exception as e:
            print(f"Erro inesperado ao buscar dados históricos: {e}")
            messagebox.showerror("Erro", f"Ocorreu um erro inesperado ao buscar
     dados: {e}")
            return None

    if not all_klines:
        print("Nenhum dado histórico buscado.")
        return pd.DataFrame()


    df = pd.DataFrame(all_klines, columns=[
        'timestamp', 'open', 'high', 'low', 'close', 'volume',
        'close_time', 'quote_asset_volume', 'number_of_trades',
        'taker_buy_base_asset_volume', 'taker_buy_quote_asset_volume', 'ignore'
    ])
    df['timestamp'] = pd.to_datetime(df['timestamp'], unit='ms')
    df.set_index('timestamp', inplace=True)

    for col in ['open', 'high', 'low', 'close', 'volume']:
        df[col] = pd.to_numeric(df[col], errors='coerce')
    df.dropna(subset=['open', 'high', 'low', 'close', 'volume'], inplace=True)


    df = df[~df.index.duplicated(keep='first')]
    df.sort_index(inplace=True)

    print(f"Busca finalizada. Total de klines processadas: {len(df)}")
    return df
```

```
def interval_to_milliseconds(interval):
    """Converte string de intervalo Binance para milissegundos."""
    multipliers = {'m': 60*1000, 'h': 60*60*1000, 'd': 24*60*60*1000, 'w':
    7*24*60*60*1000}
    try:
        num = int(interval[:-1])
        unit = interval[-1].lower()
        if unit in multipliers:
            return num * multipliers[unit]
    except:
        pass
    return None


class TradingBotGUI:
    def __init__(self, root):
        self.root = root
        self.root.title(f"Bot de Trading ({'Testnet' if USE_TESTNET else 'Real
    '}) - {SYMBOL}")
        self.root.geometry("1300x850")
        self.root.minsize(1000, 700)

        self.client_initialized = False
        self.full_historical_data = pd.DataFrame()
        self.display_data = pd.DataFrame()


        self.bg_color = '#181a20'
        self.fg_color = 'white'
        self.entry_bg = '#23272e'
        self.button_bg = '#23272e'
        self.button_active_bg = '#343a40'
        self.frame_bg = '#181a20'
        self.label_bg = '#181a20'
        self.plot_bg = '#181a20'
        self.plot_fg = 'white'
        self.candle_up = '#26a69a'
        self.candle_down = '#ef5350'
        self.ma_fast_color = '#1976d2'
        self.ma_slow_color = '#ffa726'
```

```
        style = ttk.Style()
        try:
            style.theme_use('clam')
        except tk.TclError:
            print("Clam theme not available, using default.")

        style.configure('.', background=self.bg_color, foreground=self.fg_color,
 fieldbackground=self.entry_bg, borderwidth=0, focuscolor=self.fg_color)
        style.configure('TLabel', background=self.label_bg, foreground=self.
fg_color, padding=5)
        style.configure('TFrame', background=self.frame_bg)
        style.configure('TButton', background=self.button_bg, foreground=self.
fg_color, padding=5, relief='flat')
        style.map('TButton', background=[('active', self.button_active_bg), ('
disabled', '#555')])
        style.configure('TEntry', fieldbackground=self.entry_bg, foreground=
self.fg_color, insertbackground=self.fg_color)
        style.configure('TCombobox', fieldbackground=self.entry_bg, foreground=
self.fg_color, background=self.button_bg, arrowcolor=self.fg_color)
        style.map('TCombobox', fieldbackground=[('readonly', self.entry_bg)],
foreground=[('readonly', self.fg_color)])
        style.configure('TLabelframe', background=self.frame_bg, foreground=
self.fg_color, relief='groove', borderwidth=1)
        style.configure('TLabelframe.Label', background=self.label_bg,
foreground=self.fg_color)
        style.configure('TRadiobutton', background=self.label_bg, foreground=
self.fg_color, indicatorcolor=self.button_bg)
        style.map('TRadiobutton', background=[('active', self.bg_color)],
indicatorcolor=[('selected', self.ma_fast_color)])

        self.root.configure(bg=self.bg_color)


        self.interval_options = {
            "1 min": Client.KLINE_INTERVAL_1MINUTE,
            "5 min": Client.KLINE_INTERVAL_5MINUTE,
            "15 min": Client.KLINE_INTERVAL_15MINUTE,
            "30 min": Client.KLINE_INTERVAL_30MINUTE,
```

```
            "1 hour": Client.KLINE_INTERVAL_1HOUR,
            "4 hour": Client.KLINE_INTERVAL_4HOUR,
            "1 day": Client.KLINE_INTERVAL_1DAY
        }
        self.current_interval = self.interval_options["1 hour"]
        self.is_running = False
        self.is_hibernating = False
        self.trading_thread = None
        self.hibernate_thread = None
        self.log_queue = queue.Queue()
        self.price_update_queue = queue.Queue()
        self.plot_update_queue = queue.Queue()
        self.trade_log_file = None
        self.in_position = False
        self.strategy_var = tk.StringVar(value="Order Book")
        self.ma_fast_var = tk.StringVar(value=str(DEFAULT_MA_FAST))
        self.ma_slow_var = tk.StringVar(value=str(DEFAULT_MA_SLOW))


        self.main_container = ttk.Frame(self.root, padding="10")
        self.main_container.grid(row=0, column=0, sticky="nsew")
        self.root.grid_rowconfigure(0, weight=1)
        self.root.grid_columnconfigure(0, weight=1)

        self.main_container.grid_columnconfigure(0, weight=3)
        self.main_container.grid_columnconfigure(1, weight=1)
        self.main_container.grid_rowconfigure(0, weight=1)


        self.chart_frame = ttk.LabelFrame(self.main_container, text="Gráfico de
    Preços", padding="5")
        self.chart_frame.grid(row=0, column=0, sticky="nsew", padx=(0, 10))
        self.chart_frame.grid_rowconfigure(0, weight=1)
        self.chart_frame.grid_columnconfigure(0, weight=1)


        self.fig = Figure(figsize=(10, 6), dpi=100, facecolor=self.plot_bg)
        self.ax = self.fig.add_subplot(111, facecolor=self.plot_bg)
        self.setup_plot_style()
```

```
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.chart_frame)
        self.canvas_widget = self.canvas.get_tk_widget()
        self.canvas_widget.grid(row=0, column=0, sticky="nsew")


        self.toolbar_frame = ttk.Frame(self.chart_frame)
        self.toolbar_frame.grid(row=1, column=0, sticky="ew")
        self.toolbar = NavigationToolbar2Tk(self.canvas, self.toolbar_frame,
    pack_toolbar=False)
        self.toolbar.update()
        self.toolbar.configure(background=self.button_bg)
        for button in self.toolbar.winfo_children():
            try:
                button.configure(background=self.button_bg, foreground=self.
    fg_color)
            except tk.TclError:
                button.configure(background=self.button_bg)
        self.toolbar.grid(row=0, column=0, sticky="ew")


        self.right_frame = ttk.Frame(self.main_container, padding="5")
        self.right_frame.grid(row=0, column=1, sticky="nsew")
        self.right_frame.grid_columnconfigure(0, weight=1)

        self.right_frame.grid_rowconfigure(0, weight=0)
        self.right_frame.grid_rowconfigure(1, weight=0)
        self.right_frame.grid_rowconfigure(2, weight=0)
        self.right_frame.grid_rowconfigure(3, weight=0)
        self.right_frame.grid_rowconfigure(4, weight=1)


        strategy_frame = ttk.LabelFrame(self.right_frame, text="Estratégia de
    Trading", padding="10")
        strategy_frame.grid(row=0, column=0, sticky="new", pady=(0, 10))
        strategy_frame.grid_columnconfigure(0, weight=1)
        strategy_frame.grid_columnconfigure(1, weight=1)

        self.orderbook_radio = ttk.Radiobutton(strategy_frame, text="Livro de
    Ordens", variable=self.strategy_var, value="Order Book", command=self.
    on_strategy_change)
```

```
        self.orderbook_radio.grid(row=0, column=0, sticky="w", padx=5)
        self.crossover_radio = ttk.Radiobutton(strategy_frame, text="Cruzamento
     MA", variable=self.strategy_var, value="MA Crossover", command=self.
    on_strategy_change)
        self.crossover_radio.grid(row=0, column=1, sticky="w", padx=5)


        self.settings_frame = ttk.Frame(self.right_frame)
        self.settings_frame.grid(row=1, column=0, sticky="new", pady=(0, 10))
        self.settings_frame.grid_columnconfigure(0, weight=1)


        self.orderbook_settings_frame = ttk.LabelFrame(self.settings_frame,
    text="Configurações de Livro de Ordens", padding="10")
        self.orderbook_settings_frame.grid(row=0, column=0, sticky="new")
        self.orderbook_settings_frame.grid_columnconfigure(1, weight=1)

        ttk.Label(self.orderbook_settings_frame, text="Intervalo:").grid(row=0,
     column=0, sticky="w", padx=5, pady=2)
        self.interval_var = tk.StringVar(value="1 hour")
        self.interval_combo = ttk.Combobox(self.orderbook_settings_frame,
    textvariable=self.interval_var,
                                           values=list(self.interval_options.keys
    ()),
                                           state="readonly", width=10)
        self.interval_combo.grid(row=0, column=1, sticky="ew", padx=5, pady=2)
        self.interval_combo.bind('<<ComboboxSelected>>', self.
    on_interval_change)

        ttk.Label(self.orderbook_settings_frame, text="Velas:").grid(row=1,
    column=0, sticky="w", padx=5, pady=2)
        self.periods_to_display_var = tk.StringVar(value="200")
        self.periods_to_display_entry = ttk.Entry(self.orderbook_settings_frame,
     textvariable=self.periods_to_display_var, width=10)
        self.periods_to_display_entry.grid(row=1, column=1, sticky="ew", padx=5,
     pady=2)
        self.periods_to_display_entry.bind("<Return>", self.on_periods_change)
        self.periods_to_display_entry.bind("<FocusOut>", self.on_periods_change
    )
```

```
        ttk.Label(self.orderbook_settings_frame, text="Preço para Comprar:").
    grid(row=2, column=0, sticky="w", padx=5, pady=2)
        self.buy_price_var = tk.StringVar(value="")
        self.buy_price_entry = ttk.Entry(self.orderbook_settings_frame,
    textvariable=self.buy_price_var)
        self.buy_price_entry.grid(row=2, column=1, sticky="ew", padx=5, pady=2)

        ttk.Label(self.orderbook_settings_frame, text="Preço para Vender:").
    grid(row=3, column=0, sticky="w", padx=5, pady=2)
        self.sell_price_var = tk.StringVar(value="")
        self.sell_price_entry = ttk.Entry(self.orderbook_settings_frame,
    textvariable=self.sell_price_var)
        self.sell_price_entry.grid(row=3, column=1, sticky="ew", padx=5, pady
    =2)



        self.crossover_settings_frame = ttk.LabelFrame(self.settings_frame,
    text="Configurações de Cruzamento MA", padding="10")

        self.crossover_settings_frame.grid_columnconfigure(1, weight=1)

        ttk.Label(self.crossover_settings_frame, text="Intervalo:").grid(row=0,
     column=0, sticky="w", padx=5, pady=2)

        self.interval_combo_crossover = ttk.Combobox(self.
    crossover_settings_frame, textvariable=self.interval_var,
                                                     values=list(self.
    interval_options.keys()),
                                                     state="readonly", width=10)
        self.interval_combo_crossover.grid(row=0, column=1, sticky="ew", padx=5,
     pady=2)
        self.interval_combo_crossover.bind('<<ComboboxSelected>>', self.
    on_interval_change)

        ttk.Label(self.crossover_settings_frame, text="Velas:").grid(row=1,
    column=0, sticky="w", padx=5, pady=2)

        self.periods_to_display_entry_crossover = ttk.Entry(self.
    crossover_settings_frame, textvariable=self.periods_to_display_var, width
    =10)
```

```
        self.periods_to_display_entry_crossover.grid(row=1, column=1, sticky="
    ew", padx=5, pady=2)
        self.periods_to_display_entry_crossover.bind("<Return>", self.
    on_periods_change)
        self.periods_to_display_entry_crossover.bind("<FocusOut>", self.
    on_periods_change)

        ttk.Label(self.crossover_settings_frame, text="MA Rápida:").grid(row=2,
     column=0, sticky="w", padx=5, pady=2)
        self.ma_fast_entry = ttk.Entry(self.crossover_settings_frame,
    textvariable=self.ma_fast_var, width=10)
        self.ma_fast_entry.grid(row=2, column=1, sticky="ew", padx=5, pady=2)
        self.ma_fast_entry.bind("<Return>", self.on_ma_period_change)
        self.ma_fast_entry.bind("<FocusOut>", self.on_ma_period_change)

        ttk.Label(self.crossover_settings_frame, text="MA Lenta:").grid(row=3,
    column=0, sticky="w", padx=5, pady=2)
        self.ma_slow_entry = ttk.Entry(self.crossover_settings_frame,
    textvariable=self.ma_slow_var, width=10)
        self.ma_slow_entry.grid(row=3, column=1, sticky="ew", padx=5, pady=2)
        self.ma_slow_entry.bind("<Return>", self.on_ma_period_change)
        self.ma_slow_entry.bind("<FocusOut>", self.on_ma_period_change)


        market_frame = ttk.LabelFrame(self.right_frame, text="Informações de
    Mercado", padding="10")
        market_frame.grid(row=2, column=0, sticky="new", pady=(0, 10))
        market_frame.grid_columnconfigure(0, weight=1)

        self.current_price_label = ttk.Label(market_frame, text="Preço Atual:
    --", font=("TkDefaultFont", 12))
        self.current_price_label.grid(row=0, column=0, sticky="w", padx=5, pady
    =5)



        status_button_frame = ttk.Frame(self.right_frame)
        status_button_frame.grid(row=3, column=0, sticky="new", pady=(0, 10))
        status_button_frame.grid_columnconfigure(0, weight=1)

        self.status_label = ttk.Label(status_button_frame, text="Status:
```

```
Desconectado", foreground="orange", font=("TkDefaultFont", 10, "bold"))
        self.status_label.grid(row=0, column=0, columnspan=3, sticky="ew", pady
=(0, 5))

        button_subframe = ttk.Frame(status_button_frame)
        button_subframe.grid(row=1, column=0, sticky="ew")
        button_subframe.grid_columnconfigure(0, weight=1)
        button_subframe.grid_columnconfigure(1, weight=1)
        button_subframe.grid_columnconfigure(2, weight=1)

        self.start_button = ttk.Button(button_subframe, text="Iniciar Trading",
 command=self.start_trading, state='disabled')
        self.start_button.grid(row=0, column=0, padx=5, pady=5, sticky='ew')

        self.hibernate_button = ttk.Button(button_subframe, text="Hibernar",
command=self.toggle_hibernate, state='disabled')
        self.hibernate_button.grid(row=0, column=1, padx=5, pady=5, sticky='ew
')

        self.stop_button = ttk.Button(button_subframe, text="Parar", command=
self.stop_all, state='disabled')
        self.stop_button.grid(row=0, column=2, padx=5, pady=5, sticky='ew')


        terminal_frame = ttk.LabelFrame(self.right_frame, text="Log", padding
="5")
        terminal_frame.grid(row=4, column=0, sticky="nsew")
        terminal_frame.grid_rowconfigure(0, weight=1)
        terminal_frame.grid_columnconfigure(0, weight=1)

        self.terminal = scrolledtext.ScrolledText(terminal_frame, wrap=tk.WORD,
 height=10,
                                                  bg=self.entry_bg, fg=self.
fg_color,
                                                  insertbackground=self.
fg_color, relief='flat', borderwidth=0)
        self.terminal.grid(row=0, column=0, sticky="nsew")
        self.terminal.config(state='disabled')
```

```
        self.log_message("GUI Inicializada. Aguardando conexão com API...")
        self.update_log_display()
        self.update_price_from_queue()
        self.update_plot_from_queue()
        self.on_strategy_change()

        self.root.after(100, self.initialize_and_load)

    def setup_plot_style(self):
        """Configura o estilo dos eixos Matplotlib para tema escuro."""
        self.ax.clear()
        self.ax.set_facecolor(self.plot_bg)
        self.ax.tick_params(axis='x', colors=self.plot_fg)
        self.ax.tick_params(axis='y', colors=self.plot_fg)
        self.ax.xaxis.label.set_color(self.plot_fg)
        self.ax.yaxis.label.set_color(self.plot_fg)
        self.ax.title.set_color(self.plot_fg)
        for spine in self.ax.spines.values():
            spine.set_edgecolor(self.plot_fg)
        self.ax.grid(color=self.entry_bg, linestyle='--', linewidth=0.5)
        self.fig.patch.set_facecolor(self.plot_bg)
        self.fig.tight_layout(pad=2.0)

    def initialize_and_load(self):
        """Inicializa cliente e carrega dados iniciais."""
        self.client_initialized = initialize_binance_client()
        if self.client_initialized:
            self.status_label.config(text="Status: Ocioso", foreground="yellow
    ")
            self.update_ui_state('stopped')
            self.log_message("Cliente Binance conectado. Carregando dados
    iniciais...")
            self.load_historical_data()
            self.start_price_monitor()
        else:
            self.status_label.config(text="Status: Erro da API", foreground="
    red")
            self.update_ui_state('error')
            self.log_message("Falha ao inicializar cliente Binance. Verifique
    chaves da API e conexão.")
```

```
    def start_price_monitor(self):
        """Inicia uma thread para buscar continuamente o preço atual."""
        if not self.client_initialized:
            return
        price_thread = threading.Thread(target=self._price_monitor_loop, daemon
    =True)
        price_thread.start()

    def _price_monitor_loop(self):
        """Busca continuamente o preço atual e coloca na fila."""
        while True:
            if client:
                price = get_current_price(SYMBOL)
                if price is not None:
                    self.price_update_queue.put(price)
                else:

                    pass
            else:

                self.log_message("Monitor de preço parado: Cliente inválido.")
                break
            time.sleep(5)

    def update_price_from_queue(self):
        """Processa atualizações de preço da fila na thread principal."""
        try:
            while True:
                price = self.price_update_queue.get_nowait()
                self.current_price_label.config(text=f"Preço Atual: {price:.2f
    }")
        except queue.Empty:
            pass
        finally:
            self.root.after(500, self.update_price_from_queue)

    def update_plot_from_queue(self):
        """Processa atualizações de gráfico da fila na thread principal."""
        try:
```

```
            while True:
                data = self.plot_update_queue.get_nowait()
                if data is None:
                    self._clear_plot()
                elif isinstance(data, pd.DataFrame):
                    self.full_historical_data = data
                    self._update_display_data()
                    self._draw_plot()
                elif data == "redraw":
                    self._draw_plot()

        except queue.Empty:
            pass
        finally:
            self.root.after(500, self.update_plot_from_queue)

    def on_strategy_change(self):
        """Lida com a mudança entre estratégias de trading na GUI."""
        selected_strategy = self.strategy_var.get()
        self.log_message(f"Estratégia alterada para: {selected_strategy}")

        if selected_strategy == "Order Book":
            self.crossover_settings_frame.grid_remove()
            self.orderbook_settings_frame.grid(row=0, column=0, sticky="new")

            self.interval_combo.grid(row=0, column=1, sticky="ew", padx=5, pady
    =2)
            self.periods_to_display_entry.grid(row=1, column=1, sticky="ew",
    padx=5, pady=2)
        elif selected_strategy == "MA Crossover":
            self.orderbook_settings_frame.grid_remove()
            self.crossover_settings_frame.grid(row=0, column=0, sticky="new")

            self.interval_combo_crossover.grid(row=0, column=1, sticky="ew",
    padx=5, pady=2)
            self.periods_to_display_entry_crossover.grid(row=1, column=1,
    sticky="ew", padx=5, pady=2)


        self.plot_update_queue.put("redraw")
```

Versão Final Homologada
27/12/2025 13:45

```
    def on_interval_change(self, event=None):
        """Lida com mudança de intervalo, recarrega dados se o bot estiver
    parado."""
        if self.is_running or self.is_hibernating:
            messagebox.showwarning("Aviso", "Por favor, pare o bot antes de
    alterar o intervalo.")
            for name, val in self.interval_options.items():
                if val == self.current_interval:
                    self.interval_var.set(name)
                    break
            return
        new_interval_name = self.interval_var.get()
        self.current_interval = self.interval_options[new_interval_name]
        self.log_message(f"Intervalo alterado para: {new_interval_name}.
    Recarregando dados...")
        self.load_historical_data()

    def on_periods_change(self, event=None):
        """Lida com mudança no número de velas a exibir no gráfico."""
        if self._update_display_data():
             self.plot_update_queue.put("redraw")

    def on_ma_period_change(self, event=None):
        """Lida com mudança nos períodos de MA, redesenhá o gráfico."""
        if self.validate_ma_periods():
            self.log_message(f"Períodos de MA atualizados: Rápida={self.
    ma_fast_var.get()}, Lenta={self.ma_slow_var.get()}")
            self.plot_update_queue.put("redraw")

    def _update_display_data(self):
        """Atualiza self.display_data fatiando self.full_historical_data."""
        if self.full_historical_data.empty:
            self.display_data = pd.DataFrame()
            return True

        try:
            periods = int(self.periods_to_display_var.get())
            if periods <= 5:
                raise ValueError("Número de velas a exibir deve ser maior que
```

```
    5.")

            self.display_data = self.full_historical_data.iloc[-periods:].copy
    ()
            self.log_message(f"Exibindo últimas {len(self.display_data)} velas
    .")
            return True
        except ValueError as e:
            messagebox.showerror("Entrada Inválida", f"Velas a exibir: {e}")

            try:
                current_periods = len(self.display_data) if not self.
    display_data.empty else 200
                self.periods_to_display_var.set(str(current_periods))
            except:
                 self.periods_to_display_var.set("200")
            return False

    def load_historical_data(self):
        """Carrega TODOS os dados históricos usando a nova função e coloca na
    fila de gráfico."""
        if not self.client_initialized:
            self.log_message("Não é possível carregar dados: Cliente não
    inicializado.")
            return

        self.log_message(f"Carregando TODOS os dados históricos para {SYMBOL}
    ({self.interval_var.get()}) de {DEFAULT_START_DATE_STR}...")
        self.plot_update_queue.put(None)


        df = get_all_historical_data(SYMBOL, self.current_interval,
    DEFAULT_START_DATE_STR)

        if df is not None:
            if not df.empty:
                self.plot_update_queue.put(df)
                self.log_message(f"Carregados com sucesso {len(df)} pontos de
    dados históricos totais.")
            else:
```

```
                self.log_message("Nenhum dado histórico retornado da API.")
                self.plot_update_queue.put(pd.DataFrame())
        else:

            self.log_message("Falha ao carregar dados históricos.")
            self.plot_update_queue.put(pd.DataFrame())

    def _clear_plot(self):
        """Limpa os eixos Matplotlib e define o estilo."""
        self.setup_plot_style()
        self.ax.set_title(f"Gráfico de Preços {SYMBOL} ({self.interval_var.get
    ()})", color=self.plot_fg)
        self.ax.set_xlabel("Tempo", color=self.plot_fg)
        self.ax.set_ylabel("Preço", color=self.plot_fg)
        self.canvas.draw_idle()

    def _draw_plot(self):
        """Desenha o gráfico de preços usando self.display_data."""
        self._clear_plot()

        if self.display_data.empty:
            self.log_message('Nenhum dado disponível para exibir no gráfico.')
            self.canvas.draw_idle()
            return

        df_plot = self.display_data

        if len(df_plot) < 2:
            self.log_message('Pontos de dados insuficientes na janela de exibiç
    ão para plotar velas.')
            self.canvas.draw_idle()
            return



        try:
            ma_fast_period = int(self.ma_fast_var.get())
            ma_slow_period = int(self.ma_slow_var.get())
            if ma_fast_period <= 0 or ma_slow_period <= 0 or ma_fast_period >=
    ma_slow_period:
```

```
                raise ValueError("Períodos de MA inválidos para plotagem.")
        except ValueError:

            ma_fast_period = DEFAULT_MA_FAST
            ma_slow_period = DEFAULT_MA_SLOW
            self.log_message(f"Períodos de MA inválidos na GUI, usando padrões
    ({ma_fast_period}/{ma_slow_period}) para o gráfico.")

        df_plot[f'MA{ma_fast_period}'] = df_plot['close'].rolling(window=
    ma_fast_period).mean()
        df_plot[f'MA{ma_slow_period}'] = df_plot['close'].rolling(window=
    ma_slow_period).mean()



        width = (mdates.date2num(df_plot.index[1]) - mdates.date2num(df_plot.
    index[0])) * 0.7
        width2 = width * 0.1

        up = df_plot[df_plot.close >= df_plot.open]
        down = df_plot[df_plot.close < df_plot.open]

        self.ax.bar(up.index, up.close - up.open, width, bottom=up.open, color=
    self.candle_up, alpha=0.9)
        self.ax.bar(up.index, up.high - up.close, width2, bottom=up.close,
    color=self.candle_up, alpha=0.9)
        self.ax.bar(up.index, up.low - up.open, width2, bottom=up.open, color=
    self.candle_up, alpha=0.9)

        self.ax.bar(down.index, down.close - down.open, width, bottom=down.open,
     color=self.candle_down, alpha=0.9)
        self.ax.bar(down.index, down.high - down.open, width2, bottom=down.open,
     color=self.candle_down, alpha=0.9)
        self.ax.bar(down.index, down.low - down.close, width2, bottom=down.
    close, color=self.candle_down, alpha=0.9)



        self.ax.plot(df_plot.index, df_plot[f'MA{ma_fast_period}'], color=self.
    ma_fast_color, linewidth=1.5, label=f'MA{ma_fast_period}')
        self.ax.plot(df_plot.index, df_plot[f'MA{ma_slow_period}'], color=self.
```

```
    ma_slow_color, linewidth=1.5, label=f'MA{ma_slow_period}')



        time_range_days = (df_plot.index[-1] - df_plot.index[0]).days
        if time_range_days > 30:
            date_format = '%Y-%m-%d'
        elif time_range_days > 1:
            date_format = '%m-%d %H:%M'
        else:
            date_format = '%H:%M:%S'
        self.ax.xaxis.set_major_formatter(mdates.DateFormatter(date_format))
        self.ax.xaxis.set_major_locator(mticker.MaxNLocator(12))
        self.ax.legend(loc='upper left', fontsize='small', facecolor=self.
    entry_bg, edgecolor=self.fg_color, labelcolor=self.fg_color)
        self.ax.margins(x=0.01, y=0.05)
        self.fig.autofmt_xdate()

        self.canvas.draw_idle()
        self.log_message(f"Gráfico atualizado. Exibindo {len(df_plot)} de {len(
    self.full_historical_data)} velas totais.")

    def log_message(self, message):
        """Adiciona uma mensagem com timestamp à fila de log."""
        now = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        self.log_queue.put(f"[{now}] {message}")

    def update_log_display(self):
        """Atualiza a exibição do log da fila na thread principal."""
        try:
            while True:
                message = self.log_queue.get_nowait()
                self.terminal.config(state='normal')
                self.terminal.insert(tk.END, message + "\n")
                self.terminal.see(tk.END)
                self.terminal.config(state='disabled')
        except queue.Empty:
            pass
        finally:
            self.root.after(100, self.update_log_display)
```

```
    def validate_orderbook_settings(self):
        """Valida as entradas de configuração do livro de ordens (preços-alvo)
    ."""
        try:
            buy_price_str = self.buy_price_var.get()
            sell_price_str = self.sell_price_var.get()

            if not buy_price_str and not sell_price_str:
                messagebox.showerror("Erro de Entrada", "Livro de Ordens: Por
    favor, insira pelo menos um preço (compra ou venda).")
                return False

            if buy_price_str:
                buy_price = float(buy_price_str)
                if buy_price <= 0:
                    messagebox.showerror("Erro de Entrada", "Livro de Ordens:
    Preço de compra deve ser positivo.")
                    return False

            if sell_price_str:
                sell_price = float(sell_price_str)
                if sell_price <= 0:
                    messagebox.showerror("Erro de Entrada", "Livro de Ordens:
    Preço de venda deve ser positivo.")
                    return False


            if buy_price_str and sell_price_str:
                buy_price = float(buy_price_str)
                sell_price = float(sell_price_str)
                if buy_price >= sell_price:
                    messagebox.showerror("Erro de Lógica", "Livro de Ordens:
    Preço de compra deve ser menor que preço de venda.")
                    return False

            return True
        except ValueError:
            messagebox.showerror("Erro de Entrada", "Livro de Ordens: Por favor,
     insira números válidos para os preços.")
```

```
            return False

    def validate_ma_periods(self):
        """Valida as entradas de períodos de MA."""
        try:
            fast_period_str = self.ma_fast_var.get()
            slow_period_str = self.ma_slow_var.get()
            if not fast_period_str or not slow_period_str:
                 messagebox.showerror("Erro de Entrada", "Cruzamento MA: Por
    favor, insira períodos de MA Rápida e Lenta.")
                 return False
            fast_period = int(fast_period_str)
            slow_period = int(slow_period_str)
            if fast_period <= 0 or slow_period <= 0:
                messagebox.showerror("Erro de Entrada", "Cruzamento MA: Perí
    odos devem ser inteiros positivos.")
                return False
            if fast_period >= slow_period:
                messagebox.showerror("Erro de Lógica", "Cruzamento MA: Período
    da MA Rápida deve ser menor que o da MA Lenta.")
                return False



            return True
        except ValueError:
            messagebox.showerror("Erro de Entrada", "Cruzamento MA: Por favor,
    insira inteiros válidos para períodos de MA.")
            return False

    def update_ui_state(self, state):
        """Updates the UI elements based on the bot's state."""
        is_client_ok = self.client_initialized
        current_strategy = self.strategy_var.get()

        if state == 'running':
            status_text, status_color = "Status: Running", "green"
            start_state, hibernate_state, stop_state = 'disabled', 'disabled', '
    normal'
            interval_state, periods_state = 'disabled', 'disabled'
```

```
            orderbook_buy_state, orderbook_sell_state = 'disabled', 'disabled'
            ma_fast_state, ma_slow_state = 'disabled', 'disabled'
            strategy_radios_state = 'disabled'
        elif state == 'hibernating':
            status_text, status_color = "Status: Hibernating", "blue"
            start_state, hibernate_state, stop_state = 'disabled', 'normal', '
    normal'
            self.hibernate_button.config(text="Wake Up")
            interval_state, periods_state = 'disabled', 'disabled'
            orderbook_buy_state, orderbook_sell_state = 'disabled', 'disabled'
            ma_fast_state, ma_slow_state = 'disabled', 'disabled'
            strategy_radios_state = 'disabled'
        elif state == 'stopped':
            status_text, status_color = "Status: Idle", "yellow"
            start_state = 'normal' if is_client_ok else 'disabled'
            hibernate_state = 'normal' if is_client_ok else 'disabled'
            stop_state = 'disabled'
            self.hibernate_button.config(text="Hibernate")
            interval_state, periods_state = 'readonly', 'normal'
            orderbook_buy_state = 'normal' if current_strategy == "Order Book"
    else 'disabled'
            orderbook_sell_state = 'normal' if current_strategy == "Order Book"
     else 'disabled'
            ma_fast_state = 'normal' if current_strategy == "MA Crossover" else
     'disabled'
            ma_slow_state = 'normal' if current_strategy == "MA Crossover" else
     'disabled'
            strategy_radios_state = 'normal'
        elif state == 'error':
            status_text, status_color = "Status: API Error", "red"
            start_state, hibernate_state, stop_state = 'disabled', 'disabled', '
    disabled'
            interval_state, periods_state = 'disabled', 'disabled'
            orderbook_buy_state, orderbook_sell_state = 'disabled', 'disabled'
            ma_fast_state, ma_slow_state = 'disabled', 'disabled'
            strategy_radios_state = 'disabled'
        else:
            status_text, status_color = "Status: Disconnected", "orange"
            start_state, hibernate_state, stop_state = 'disabled', 'disabled', '
    disabled'
```

```
            interval_state, periods_state = 'disabled', 'disabled'
            orderbook_buy_state, orderbook_sell_state = 'disabled', 'disabled'
            ma_fast_state, ma_slow_state = 'disabled', 'disabled'
            strategy_radios_state = 'disabled'


        self.status_label.config(text=status_text, foreground=status_color)
        self.start_button.config(state=start_state)
        self.hibernate_button.config(state=hibernate_state)
        self.stop_button.config(state=stop_state)
        self.interval_combo.config(state=interval_state)
        self.interval_combo_crossover.config(state=interval_state)
        self.periods_to_display_entry.config(state=periods_state)
        self.periods_to_display_entry_crossover.config(state=periods_state)
        self.buy_price_entry.config(state=orderbook_buy_state)
        self.sell_price_entry.config(state=orderbook_sell_state)
        self.ma_fast_entry.config(state=ma_fast_state)
        self.ma_slow_entry.config(state=ma_slow_state)
        self.orderbook_radio.config(state=strategy_radios_state)
        self.crossover_radio.config(state=strategy_radios_state)

    def start_trading(self):
        """Starts the trading bot thread based on the selected strategy."""
        if not self.client_initialized:
            self.log_message("Cannot start: Binance client not initialized.")
            return
        if self.is_running or self.is_hibernating:
            self.log_message("Bot is already running or hibernating.")
            return

        selected_strategy = self.strategy_var.get()
        valid_settings = False
        if selected_strategy == "Order Book":
            valid_settings = self.validate_orderbook_settings()
            if valid_settings:
                self.log_message(f"Starting trading with Order Book strategy (
    agendamento de operações)...")
                buy_price_str = self.buy_price_var.get()
                sell_price_str = self.sell_price_var.get()
                if buy_price_str:
```

```
                self.log_message(f"Ordem de compra agendada quando preço
atingir: {buy_price_str}")
            if sell_price_str:
                self.log_message(f"Ordem de venda agendada quando preço
atingir: {sell_price_str}")
    elif selected_strategy == "MA Crossover":
        valid_settings = self.validate_ma_periods()
        if valid_settings:

             slow_ma = int(self.ma_slow_var.get())
             if len(self.full_historical_data) < slow_ma:
                 messagebox.showerror("Data Error", f"Not enough historical
 data ({len(self.full_historical_data)} candles) for the slow MA ({slow_ma})
. Load more data or use a shorter period.")
                 valid_settings = False
             else:
                self.log_message(f"Starting trading with MA Crossover
strategy (Fast: {self.ma_fast_var.get()}, Slow: {self.ma_slow_var.get()})
...")

    if not valid_settings:
        self.log_message("Cannot start trading due to invalid settings.")
        return

    self.is_running = True
    self.is_hibernating = False
    self.in_position = False
    self.update_ui_state('running')


    now_str = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    self.trade_log_file = f'trade_log_{SYMBOL}_{selected_strategy.replace("
 ", "")}_{now_str}.csv'
    try:
        with open(self.trade_log_file, 'w', newline='') as csvfile:
            writer = csv.writer(csvfile)

            if selected_strategy == "Order Book":
                writer.writerow(['Timestamp', 'Action', 'Strategy', 'Price',
 'Target Price', 'Best Bid', 'Best Ask', 'Quantity', 'Order ID', 'Status'])
```

```
                else:
                    writer.writerow(['Timestamp', 'Action', 'Strategy', 'Price',
 'Fast MA', 'Slow MA', 'Quantity', 'Order ID', 'Status'])
            self.log_message(f"Trade log initialized: {self.trade_log_file}")
        except IOError as e:
            self.log_message(f"Error creating trade log file: {e}")
            self.trade_log_file = None

        self.trading_thread = threading.Thread(target=self._trade_bot_loop,
daemon=True)
        self.trading_thread.start()

    def _trade_bot_loop(self):
        """The main loop for the trading logic, adapts based on selected
strategy."""
        selected_strategy = self.strategy_var.get()
        self.log_message(f"Trading loop started with strategy: {
selected_strategy}")


        if selected_strategy == "Order Book":

            buy_price_target = None
            sell_price_target = None
            buy_price_str = self.buy_price_var.get()
            sell_price_str = self.sell_price_var.get()
            if buy_price_str:
                buy_price_target = float(buy_price_str)
            if sell_price_str:
                sell_price_target = float(sell_price_str)

            last_log_time = 0
        elif selected_strategy == "MA Crossover":
            ma_fast_period = int(self.ma_fast_var.get())
            ma_slow_period = int(self.ma_slow_var.get())

            if self.full_historical_data.empty or len(self.full_historical_data
) < ma_slow_period:
                self.log_message("Error: Not enough historical data for MA
calculation in trading loop. Stopping.")
```

```
                self.root.after(0, self.stop_all)
                return
            local_data = self.full_historical_data.copy()
            local_data[f'MA{ma_fast_period}'] = local_data['close'].rolling(
    window=ma_fast_period).mean()
            local_data[f'MA{ma_slow_period}'] = local_data['close'].rolling(
    window=ma_slow_period).mean()

        while self.is_running:

            current_price = get_current_price(SYMBOL)
            if current_price is None:
                self.log_message("Failed to get current price. Retrying...")
                time.sleep(10)
                continue




            if selected_strategy == "MA Crossover" and self._should_update_data
    ():
                self.log_message("Updating historical data for MA calculation
    ...")


                updated_data = get_all_historical_data(SYMBOL, self.
    current_interval, DEFAULT_START_DATE_STR)
                if updated_data is not None and not updated_data.empty:
                    self.full_historical_data = updated_data
                    local_data = self.full_historical_data.copy()
                    local_data[f'MA{ma_fast_period}'] = local_data['close'].
    rolling(window=ma_fast_period).mean()
                    local_data[f'MA{ma_slow_period}'] = local_data['close'].
    rolling(window=ma_slow_period).mean()
                    self.log_message("Historical data updated for MAs.")

                    self.plot_update_queue.put(self.full_historical_data.copy()
    )
                else:
                    self.log_message("Failed to update historical data for MAs.
```

```
    Using previous data.")


            action = None
            order_result = None
            status = "-"
            fast_ma_val = np.nan
            slow_ma_val = np.nan
            best_bid = 0.0
            best_ask = 0.0
            imbalance_pct = 0.0

            try:
                if selected_strategy == "Order Book":



                    order_book = get_order_book(SYMBOL, limit=5)
                    if order_book is None:
                        self.log_message("Failed to get order book. Retrying
...")
                        time.sleep(5)
                        continue


                    best_bid = order_book['bids'][0][0] if order_book['bids']
else current_price
                    best_ask = order_book['asks'][0][0] if order_book['asks']
else current_price



                    price_reference = current_price


                    if buy_price_target is not None and not self.in_position:

                        if price_reference <= buy_price_target:
                            self.log_message(f"Order Book: Preço atual {
price_reference:.2f} <= Preço-alvo de compra {buy_price_target:.2f}.
```

```
    EXECUTANDO COMPRA.")
                            action = 'BUY'
                            order_result = place_buy_order(SYMBOL,
    TRADE_QUANTITY, self)
                        else:

                            current_time = time.time()
                            if current_time - last_log_time >= 30:
                                diff = buy_price_target - price_reference
                                self.log_message(f"Order Book: Aguardando
    compra. Preço atual: {price_reference:.2f}, Alvo: {buy_price_target:.2f} (
    faltam {diff:.2f})")
                                last_log_time = current_time


                    if sell_price_target is not None and self.in_position:

                        if price_reference >= sell_price_target:
                            self.log_message(f"Order Book: Preço atual {
    price_reference:.2f} >= Preço-alvo de venda {sell_price_target:.2f}.
    EXECUTANDO VENDA.")
                            action = 'SELL'
                            order_result = place_sell_order(SYMBOL,
    TRADE_QUANTITY, self)
                        else:

                            current_time = time.time()
                            if current_time - last_log_time >= 30:
                                diff = sell_price_target - price_reference
                                self.log_message(f"Order Book: Aguardando venda.
     Preço atual: {price_reference:.2f}, Alvo: {sell_price_target:.2f} (faltam {
    diff:.2f})")
                                last_log_time = current_time


                    if (not self.in_position and buy_price_target is None) or (
    self.in_position and sell_price_target is None):
                        current_time = time.time()
                        if current_time - last_log_time >= 60:
                            self.log_message(f"Order Book: Monitorando preço
```

```
    atual: {price_reference:.2f} (Melhor Bid: {best_bid:.2f}, Melhor Ask: {
    best_ask:.2f})")
                            last_log_time = current_time

                elif selected_strategy == "MA Crossover":

                    if len(local_data) >= ma_slow_period + 1:

                        last_row = local_data.iloc[-1]
                        prev_row = local_data.iloc[-2]

                        fast_ma_val = last_row[f'MA{ma_fast_period}']
                        slow_ma_val = last_row[f'MA{ma_slow_period}']
                        prev_fast_ma = prev_row[f'MA{ma_fast_period}']
                        prev_slow_ma = prev_row[f'MA{ma_slow_period}']


                        if not pd.isna(fast_ma_val) and not pd.isna(slow_ma_val
    ) and \
                           not pd.isna(prev_fast_ma) and not pd.isna(
    prev_slow_ma):


                            if not self.in_position and prev_fast_ma <=
    prev_slow_ma and fast_ma_val > slow_ma_val:
                                self.log_message(f"MA Crossover: Fast ({
    fast_ma_val:.2f}) crossed ABOVE Slow ({slow_ma_val:.2f}). BUY.")
                                action = 'BUY'
                                order_result = place_buy_order(SYMBOL,
    TRADE_QUANTITY, self)


                            elif self.in_position and prev_fast_ma >=
    prev_slow_ma and fast_ma_val < slow_ma_val:
                                self.log_message(f"MA Crossover: Fast ({
    fast_ma_val:.2f}) crossed BELOW Slow ({slow_ma_val:.2f}). SELL.")
                                action = 'SELL'
                                order_result = place_sell_order(SYMBOL,
    TRADE_QUANTITY, self)
                        else:
```

```
                        self.log_message("Waiting for MA values to
calculate...")
                else:
                    self.log_message("Waiting for enough data to calculate
MAs...")


            if action:
                if order_result and order_result.get('status') == 'FILLED':
                    status = 'FILLED'
                    if action == 'BUY':
                        self.in_position = True
                    elif action == 'SELL':
                        self.in_position = False
                    self.log_message(f"Successfully {action} {
TRADE_QUANTITY} {SYMBOL}. Status: {status}")
                else:
                    status = order_result.get('status', 'FAILED') if
order_result else 'FAILED'
                    self.log_message(f"{action} order failed or not filled.
 Status: {status}")


                if self.trade_log_file:
                    try:
                        with open(self.trade_log_file, 'a', newline='') as
csvfile:
                            writer = csv.writer(csvfile)
                            timestamp = datetime.datetime.now().strftime("%
Y-%m-%d %H:%M:%S")
                            order_id = order_result.get('orderId', 'N/A')
if order_result else 'N/A'

                            if selected_strategy == "Order Book":

                                target_price = None
                                if action == 'BUY' and buy_price_target is
not None:
                                    target_price = buy_price_target
                                elif action == 'SELL' and sell_price_target
```

```
 is not None:
                                    target_price = sell_price_target
                                writer.writerow([timestamp, action,
selected_strategy, current_price,
                                                 f"{target_price:.2f}" if
target_price is not None else "N/A",
                                                 f"{best_bid:.2f}", f"{
best_ask:.2f}",
                                                 TRADE_QUANTITY, order_id,
status])
                            else:
                                writer.writerow([timestamp, action,
selected_strategy, current_price,
                                                 f"{fast_ma_val:.2f}" if
not pd.isna(fast_ma_val) else "N/A",
                                                 f"{slow_ma_val:.2f}" if
not pd.isna(slow_ma_val) else "N/A",
                                                 TRADE_QUANTITY, order_id,
status])
                    except IOError as e:
                        self.log_message(f"Error writing to trade log: {e
}")

        except Exception as e:
            self.log_message(f"Error in trading loop: {e}")
            self.log_message(traceback.format_exc())




        if selected_strategy == "Order Book":
            time.sleep(2)
        else:
            time.sleep(10)

    self.log_message("Trading loop stopped.")

def _should_update_data(self):
    """Checks if enough time has passed to update historical data based on
```

```
    interval."""

        if self.full_historical_data is None or self.full_historical_data.empty:

            return True

        try:

            last_update_time = self.full_historical_data.index[-1].
    to_pydatetime().replace(tzinfo=None)
            current_time_utc = datetime.datetime.utcnow()
            time_diff_seconds = (current_time_utc - last_update_time).
    total_seconds()

            interval_seconds = {
                Client.KLINE_INTERVAL_1MINUTE: 60,
                Client.KLINE_INTERVAL_5MINUTE: 300,
                Client.KLINE_INTERVAL_15MINUTE: 900,
                Client.KLINE_INTERVAL_30MINUTE: 1800,
                Client.KLINE_INTERVAL_1HOUR: 3600,
                Client.KLINE_INTERVAL_4HOUR: 14400,
                Client.KLINE_INTERVAL_1DAY: 86400
            }
            required_seconds = interval_seconds.get(self.current_interval, 60)


            return time_diff_seconds >= required_seconds * 0.95
        except Exception as e:
            self.log_message(f"Error checking data update time: {e}")
            return True

    def toggle_hibernate(self):
        """Toggles the hibernation state (monitors price and chart)."""

        if not self.client_initialized:
            self.log_message("Cannot hibernate: Binance client not initialized
    .")
            return
        if self.is_running:
            self.log_message("Cannot hibernate while trading is active. Stop
```

```
    trading first.")
            return

        if not self.is_hibernating:
            self.is_hibernating = True
            self.is_running = False
            self.update_ui_state('hibernating')
            self.log_message("Entering hibernation mode (monitoring price and
    chart)...")
            self.hibernate_thread = threading.Thread(target=self.
    _hibernate_loop, daemon=True)
            self.hibernate_thread.start()
        else:
            self.is_hibernating = False
            self.update_ui_state('stopped')
            self.log_message("Exiting hibernation mode.")

    def _hibernate_loop(self):
        """Loop for hibernation mode: monitors price and updates chart."""
        self.log_message("Hibernate loop started.")
        while self.is_hibernating:


            if self._should_update_data():
                self.log_message("Hibernate: Updating historical data...")
                updated_data = get_all_historical_data(SYMBOL, self.
    current_interval, DEFAULT_START_DATE_STR)
                if updated_data is not None:
                     self.plot_update_queue.put(updated_data.copy())
            time.sleep(60)
        self.log_message("Hibernate loop stopped.")

    def stop_all(self):
        """Stops trading or hibernation."""

        stopped_action = False
        if self.is_running:
            self.log_message("Stopping trading bot...")
            self.is_running = False
            stopped_action = True
```

```
        if self.is_hibernating:
            self.log_message("Stopping hibernation...")
            self.is_hibernating = False
            stopped_action = True

        if stopped_action:


            self.update_ui_state('stopped')
            self.log_message("Bot stopped.")
        else:
            self.log_message("Bot is not running or hibernating.")

    def on_closing(self):
        """Handles window closing event."""
        if self.is_running or self.is_hibernating:
            if messagebox.askyesno("Exit Confirmation", "Bot is active. Are you
     sure you want to exit?"):
                self.stop_all()

                time.sleep(0.5)
                self.root.destroy()
            else:
                return
        else:
            self.root.destroy()


if __name__ == '__main__':
    root = tk.Tk()
    app = TradingBotGUI(root)
    root.protocol("WM_DELETE_WINDOW", app.on_closing)
    root.mainloop()
```

ANEXO B – Código do Fonte Backtest

Este apêndice apresenta o código-fonte desenvolvido pelo autor para a implementação do sistema de Backtest.

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import subprocess
import sys
import os
import threading
import json
import re
import webbrowser
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

class Config:
    class Theme:
        BG_PRIMARY = "#1e1e1e"
        BG_SECONDARY = "#2d2d2d"
        BG_TERTIARY = "#3d3d3d"
        FG_PRIMARY = "#ffffff"
        FG_SECONDARY = "#b0b0b0"
        FG_ACCENT = "#4a9eff"
        SUCCESS = "#00c853"
        ERROR = "#ff5252"
        WARNING = "#ff9800"
        INFO = "#2196f3"
        BUTTON_BG = "#3d3d3d"
        BUTTON_FG = "#ffffff"

    class Defaults:
        CRYPTO = "BTC-USD"
        INTERVAL = "1h"
```

```
        FEE = 0.0
        SLIPPAGE = 0.0
        SMA_FAST = 9
        SMA_SLOW = 20
        EMA_FAST = 9
        EMA_SLOW = 20
        LSMA_FAST = 10
        LSMA_SLOW = 50
        DATA_INICIAL = "2021-01-01"
        DATA_FINAL = ""

    class UI:
        PADDING = 10
        BUTTON_PADDING_X = 15
        BUTTON_PADDING_Y = 8
        LOG_HEIGHT = 15
        ICONS = {
            'title': '',
            'config': '',
            'strategies': '',
            'run': '',
            'stop': '',
            'folder': '',
            'log': '',
            'info': '',
            'success': '',
            'warning': '',
            'error': ''
        }

    class Fonts:
        TITLE = ("Segoe UI", 18, "bold")
        BUTTON = ("Segoe UI", 10, "bold")
        MONOSPACE = ("Consolas", 9)
        SMALL = ("Segoe UI", 9)

    WINDOW_WIDTH = 1000
    WINDOW_HEIGHT = 700
    WINDOW_MIN_WIDTH = 800
    WINDOW_MIN_HEIGHT = 600
```

```
    class Performance:
        MAX_LOG_LINES = 500
        AUTO_SCROLL = True

DarkTheme = Config.Theme

class InstavelBacktestGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Backtest de Estratégias - Crypto Trading")
        self.root.geometry(f"{Config.WINDOW_WIDTH}x{Config.WINDOW_HEIGHT}")
        self.root.minsize(Config.WINDOW_MIN_WIDTH, Config.WINDOW_MIN_HEIGHT)
        self.root.configure(bg=DarkTheme.BG_PRIMARY)

        try:
            self.root.iconbitmap("icon.ico")
        except:
            pass

        self.crypto_var = tk.StringVar(value=Config.Defaults.CRYPTO)
        self.interval_var = tk.StringVar(value=Config.Defaults.INTERVAL)
        self.fee_var = tk.DoubleVar(value=Config.Defaults.FEE)
        self.slippage_var = tk.DoubleVar(value=Config.Defaults.SLIPPAGE)
        self.data_inicial_var = tk.StringVar(value=Config.Defaults.DATA_INICIAL
    )
        self.data_final_var = tk.StringVar(value=Config.Defaults.DATA_FINAL)

        self.sma_fast_var = tk.IntVar(value=Config.Defaults.SMA_FAST)
        self.sma_slow_var = tk.IntVar(value=Config.Defaults.SMA_SLOW)
        self.ema_fast_var = tk.IntVar(value=Config.Defaults.EMA_FAST)
        self.ema_slow_var = tk.IntVar(value=Config.Defaults.EMA_SLOW)
        self.lsma_fast_var = tk.IntVar(value=Config.Defaults.LSMA_FAST)
        self.lsma_slow_var = tk.IntVar(value=Config.Defaults.LSMA_SLOW)

        self.is_running = False
        self.results_dir = None

        self.setup_styles()
        self.create_widgets()
```

```
        self.center_window()

    def setup_styles(self):
        style = ttk.Style()
        style.theme_use('clam')

        style.configure('Dark.TFrame', background=DarkTheme.BG_PRIMARY)
        style.configure('Dark.TLabelframe', background=DarkTheme.BG_PRIMARY,
    foreground=DarkTheme.FG_PRIMARY)
        style.configure('Dark.TLabelframe.Label', background=DarkTheme.
    BG_PRIMARY, foreground=DarkTheme.FG_PRIMARY)
        style.configure('Dark.TLabel', background=DarkTheme.BG_PRIMARY,
    foreground=DarkTheme.FG_PRIMARY)
        style.configure('Dark.TButton', background=DarkTheme.BUTTON_BG,
    foreground=DarkTheme.BUTTON_FG)
        style.configure('Dark.TEntry', fieldbackground=DarkTheme.BG_SECONDARY,
    foreground=DarkTheme.FG_PRIMARY)
        style.configure('Dark.TCombobox', fieldbackground=DarkTheme.
    BG_SECONDARY, foreground=DarkTheme.FG_PRIMARY)
        style.configure('Dark.TSpinbox', fieldbackground=DarkTheme.BG_SECONDARY,
     foreground=DarkTheme.FG_PRIMARY)
        style.configure('Dark.TCheckbutton', background=DarkTheme.BG_PRIMARY,
    foreground=DarkTheme.FG_PRIMARY)

    def center_window(self):
        self.root.update_idletasks()
        width = self.root.winfo_width()
        height = self.root.winfo_height()
        x = (self.root.winfo_screenwidth() // 2) - (width // 2)
        y = (self.root.winfo_screenheight() // 2) - (height // 2)
        self.root.geometry(f'{width}x{height}+{x}+{y}')

    def create_widgets(self):
        main_frame = ttk.Frame(self.root, style='Dark.TFrame')
        main_frame.pack(fill=tk.BOTH, expand=True, padx=20, pady=20)

        title_label = tk.Label(
            main_frame,
            text=f"{Config.UI.ICONS['title']} Backtest de Estratégias Crypto
    Trading",
```

Versão Final Homologada
27/12/2025 13:45

```
            font=Config.Fonts.TITLE,
            bg=DarkTheme.BG_PRIMARY,
            fg=DarkTheme.FG_ACCENT
        )
        title_label.pack(pady=(0, Config.UI.PADDING))

        config_frame = ttk.LabelFrame(main_frame, text=f"{Config.UI.ICONS['
    config']} Configurações Gerais", style='Dark.TLabelframe')
        config_frame.pack(fill=tk.X, pady=(0, Config.UI.PADDING))

        config_grid = tk.Frame(config_frame, bg=DarkTheme.BG_PRIMARY)
        config_grid.pack(fill=tk.X, padx=20, pady=20)

        tk.Label(config_grid, text="Criptomoeda:", bg=DarkTheme.BG_PRIMARY, fg=
    DarkTheme.FG_PRIMARY, font=("Segoe UI", 10)).grid(row=0, column=0, sticky=
    tk.W, padx=(0, 10), pady=5)
        crypto_entry = tk.Entry(config_grid, textvariable=self.crypto_var, bg=
    DarkTheme.BG_SECONDARY, fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10),
    width=15)
        crypto_entry.grid(row=0, column=1, padx=(0, 20), pady=5)

        tk.Label(config_grid, text="Intervalo:", bg=DarkTheme.BG_PRIMARY, fg=
    DarkTheme.FG_PRIMARY, font=("Segoe UI", 10)).grid(row=0, column=2, sticky=
    tk.W, padx=(0, 10), pady=5)
        interval_combo = ttk.Combobox(config_grid, textvariable=self.
    interval_var, values=["1m", "5m", "15m", "30m", "1h", "1d"], width=12)
        interval_combo.grid(row=0, column=3, padx=(0, 20), pady=5)

        tk.Label(config_grid, text="Taxa (%):", bg=DarkTheme.BG_PRIMARY, fg=
    DarkTheme.FG_PRIMARY, font=("Segoe UI", 10)).grid(row=0, column=4, sticky=
    tk.W, padx=(0, 10), pady=5)
        fee_entry = tk.Entry(config_grid, textvariable=self.fee_var, bg=
    DarkTheme.BG_SECONDARY, fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10),
    width=15)
        fee_entry.grid(row=0, column=5, padx=(0, 20), pady=5)

        tk.Label(config_grid, text="Derrapagem (%):", bg=DarkTheme.BG_PRIMARY,
    fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10)).grid(row=1, column=0,
    sticky=tk.W, padx=(0, 10), pady=5)
        slippage_entry = tk.Entry(config_grid, textvariable=self.slippage_var,
```

```
    bg=DarkTheme.BG_SECONDARY, fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10),
    width=15)
        slippage_entry.grid(row=1, column=1, padx=(0, 20), pady=5)

        tk.Label(config_grid, text="Data Inicial:", bg=DarkTheme.BG_PRIMARY, fg
    =DarkTheme.FG_PRIMARY, font=("Segoe UI", 10)).grid(row=1, column=2, sticky=
    tk.W, padx=(0, 10), pady=5)
        data_inicial_entry = tk.Entry(config_grid, textvariable=self.
    data_inicial_var, bg=DarkTheme.BG_SECONDARY, fg=DarkTheme.FG_PRIMARY, font
    =("Segoe UI", 10), width=15)
        data_inicial_entry.grid(row=1, column=3, padx=(0, 20), pady=5)

        tk.Label(config_grid, text="Data Final (vazio=hoje):", bg=DarkTheme.
    BG_PRIMARY, fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10)).grid(row=1,
    column=4, sticky=tk.W, padx=(0, 10), pady=5)
        data_final_entry = tk.Entry(config_grid, textvariable=self.
    data_final_var, bg=DarkTheme.BG_SECONDARY, fg=DarkTheme.FG_PRIMARY, font=("
    Segoe UI", 10), width=15)
        data_final_entry.grid(row=1, column=5, padx=(0, 20), pady=5)

        strategies_frame = ttk.LabelFrame(main_frame, text=f"{Config.UI.ICONS['
    strategies']} Configuração das Estratégias", style='Dark.TLabelframe')
        strategies_frame.pack(fill=tk.X, pady=(0, Config.UI.PADDING))

        strategies_grid = tk.Frame(strategies_frame, bg=DarkTheme.BG_PRIMARY)
        strategies_grid.pack(fill=tk.X, padx=20, pady=20)

        headers = ["Estratégia", "MA Rápida", "MA Lenta"]
        for i, header in enumerate(headers):
            tk.Label(strategies_grid, text=header, bg=DarkTheme.BG_PRIMARY, fg=
    DarkTheme.FG_ACCENT,
                     font=("Segoe UI", 10, "bold")).grid(row=0, column=i, padx
    =10, pady=5)

        tk.Label(strategies_grid, text="SMA", bg=DarkTheme.BG_PRIMARY, fg=
    DarkTheme.FG_PRIMARY,
                 font=("Segoe UI", 10)).grid(row=1, column=0, padx=10, pady=5)
        sma_fast_entry = tk.Entry(strategies_grid, textvariable=self.
    sma_fast_var, bg=DarkTheme.BG_SECONDARY,
                                  fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10),
```

```
     width=10)
        sma_fast_entry.grid(row=1, column=1, padx=10, pady=5)
        sma_slow_entry = tk.Entry(strategies_grid, textvariable=self.
    sma_slow_var, bg=DarkTheme.BG_SECONDARY,
                                  fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10),
     width=10)
        sma_slow_entry.grid(row=1, column=2, padx=10, pady=5)

        tk.Label(strategies_grid, text="EMA", bg=DarkTheme.BG_PRIMARY, fg=
    DarkTheme.FG_PRIMARY,
                 font=("Segoe UI", 10)).grid(row=2, column=0, padx=10, pady=5)
        ema_fast_entry = tk.Entry(strategies_grid, textvariable=self.
    ema_fast_var, bg=DarkTheme.BG_SECONDARY,
                                  fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10),
     width=10)
        ema_fast_entry.grid(row=2, column=1, padx=10, pady=5)
        ema_slow_entry = tk.Entry(strategies_grid, textvariable=self.
    ema_slow_var, bg=DarkTheme.BG_SECONDARY,
                                  fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10),
     width=10)
        ema_slow_entry.grid(row=2, column=2, padx=10, pady=5)

        tk.Label(strategies_grid, text="LSMA", bg=DarkTheme.BG_PRIMARY, fg=
    DarkTheme.FG_PRIMARY,
                 font=("Segoe UI", 10)).grid(row=3, column=0, padx=10, pady=5)
        lsma_fast_entry = tk.Entry(strategies_grid, textvariable=self.
    lsma_fast_var, bg=DarkTheme.BG_SECONDARY,
                                   fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10),
     width=10)
        lsma_fast_entry.grid(row=3, column=1, padx=10, pady=5)
        lsma_slow_entry = tk.Entry(strategies_grid, textvariable=self.
    lsma_slow_var, bg=DarkTheme.BG_SECONDARY,
                                   fg=DarkTheme.FG_PRIMARY, font=("Segoe UI", 10),
     width=10)
        lsma_slow_entry.grid(row=3, column=2, padx=10, pady=5)

        control_frame = tk.Frame(main_frame, bg=DarkTheme.BG_PRIMARY)
        control_frame.pack(fill=tk.X, pady=(0, Config.UI.PADDING))

        self.run_button = tk.Button(
```

```
            control_frame,
            text=f"{Config.UI.ICONS['run']} Executar Backtest",
            command=self.run_backtest,
            bg=DarkTheme.SUCCESS,
            fg=DarkTheme.FG_PRIMARY,
            font=Config.Fonts.BUTTON,
            relief=tk.FLAT,
            padx=Config.UI.BUTTON_PADDING_X,
            pady=Config.UI.BUTTON_PADDING_Y,
            cursor="hand2"
        )
        self.run_button.pack(side=tk.LEFT, padx=(0, 10))

        self.stop_button = tk.Button(
            control_frame,
            text=f"{Config.UI.ICONS['stop']} Parar",
            command=self.stop_backtest,
            bg=DarkTheme.ERROR,
            fg=DarkTheme.FG_PRIMARY,
            font=Config.Fonts.BUTTON,
            relief=tk.FLAT,
            padx=Config.UI.BUTTON_PADDING_X,
            pady=Config.UI.BUTTON_PADDING_Y,
            cursor="hand2",
            state=tk.DISABLED
        )
        self.stop_button.pack(side=tk.LEFT, padx=(0, 10))

        self.open_dashboard_button = tk.Button(
            control_frame,
            text=f" Abrir Dashboard",
            command=self.open_dashboard,
            bg=DarkTheme.INFO,
            fg=DarkTheme.FG_PRIMARY,
            font=Config.Fonts.BUTTON,
            relief=tk.FLAT,
            padx=Config.UI.BUTTON_PADDING_X,
            pady=Config.UI.BUTTON_PADDING_Y,
            cursor="hand2",
            state=tk.DISABLED
```

```
        )
        self.open_dashboard_button.pack(side=tk.LEFT, padx=(0, 10))

        self.open_folder_button = tk.Button(
            control_frame,
            text=f"{Config.UI.ICONS['folder']} Abrir Pasta",
            command=self.open_folder,
            bg=DarkTheme.WARNING,
            fg=DarkTheme.FG_PRIMARY,
            font=Config.Fonts.BUTTON,
            relief=tk.FLAT,
            padx=Config.UI.BUTTON_PADDING_X,
            pady=Config.UI.BUTTON_PADDING_Y,
            cursor="hand2"
        )
        self.open_folder_button.pack(side=tk.LEFT)

        log_frame = ttk.LabelFrame(main_frame, text=f"{Config.UI.ICONS['log']}
    Log de Execução", style='Dark.TLabelframe')
        log_frame.pack(fill=tk.BOTH, expand=True, pady=(0, Config.UI.PADDING))

        self.log_text = tk.Text(
            log_frame,
            bg=DarkTheme.BG_SECONDARY,
            fg=DarkTheme.FG_PRIMARY,
            font=Config.Fonts.MONOSPACE,
            wrap=tk.WORD,
            height=Config.UI.LOG_HEIGHT
        )
        self.log_text.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

        log_scrollbar = tk.Scrollbar(log_frame, command=self.log_text.yview)
        log_scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
        self.log_text.config(yscrollcommand=log_scrollbar.set)

        self.status_var = tk.StringVar(value="Pronto para executar backtest")
        status_bar = tk.Label(
            main_frame,
            textvariable=self.status_var,
            bg=DarkTheme.BG_TERTIARY,
```

```
            fg=DarkTheme.FG_PRIMARY,
            font=Config.Fonts.SMALL,
            relief=tk.SUNKEN,
            anchor=tk.W
        )
        status_bar.pack(fill=tk.X, pady=(10, 0))

        self.setup_button_hover_events()

    def setup_button_hover_events(self):
        buttons = [self.run_button, self.stop_button, self.
    open_dashboard_button, self.open_folder_button]

        for button in buttons:
            button.bind("<Enter>", lambda e, b=button: self.on_button_hover(b,
    True))
            button.bind("<Leave>", lambda e, b=button: self.on_button_hover(b,
    False))

    def on_button_hover(self, button, entering):
        if entering:
            if button == self.run_button:
                button.config(bg=DarkTheme.SUCCESS, relief=tk.RAISED)
            elif button == self.stop_button:
                button.config(bg=DarkTheme.ERROR, relief=tk.RAISED)
            elif button == self.open_dashboard_button:
                button.config(bg=DarkTheme.INFO, relief=tk.RAISED)
            elif button == self.open_folder_button:
                button.config(bg=DarkTheme.WARNING, relief=tk.RAISED)
        else:
            if button == self.run_button:
                button.config(bg=DarkTheme.SUCCESS, relief=tk.FLAT)
            elif button == self.stop_button:
                button.config(bg=DarkTheme.ERROR, relief=tk.FLAT)
            elif button == self.open_dashboard_button:
                button.config(bg=DarkTheme.INFO, relief=tk.FLAT)
            elif button == self.open_folder_button:
                button.config(bg=DarkTheme.WARNING, relief=tk.FLAT)

    def log_message(self, message, level="INFO"):
```

Versão Final Homologada
27/12/2025 13:45

```
        timestamp = datetime.now().strftime("%H:%M:%S")
        icon_map = {
            "INFO": Config.UI.ICONS["info"],
            "SUCCESS": Config.UI.ICONS["success"],
            "WARNING": Config.UI.ICONS["warning"],
            "ERROR": Config.UI.ICONS["error"]
        }

        icon = icon_map.get(level, Config.UI.ICONS["info"])
        self.log_text.insert(tk.END, f"[{timestamp}] {icon} {message}\n")

        lines = self.log_text.get(1.0, tk.END).split('\n')
        if len(lines) > Config.Performance.MAX_LOG_LINES:
            self.log_text.delete(1.0, f"{len(lines) - Config.Performance.
    MAX_LOG_LINES}.0")

        if Config.Performance.AUTO_SCROLL:
            self.log_text.see(tk.END)
        self.root.update_idletasks()

    def run_backtest(self):
        if self.is_running:
            messagebox.showwarning("Aviso", "Backtest já está em execução!")
            return

        self.is_running = True
        self.run_button.config(state=tk.DISABLED)
        self.stop_button.config(state=tk.NORMAL)
        self.open_dashboard_button.config(state=tk.DISABLED)
        self.status_var.set("Executando backtest...")

        self.log_text.delete(1.0, tk.END)
        self.log_message("Iniciando backtest de estratégias...", "INFO")
        self.log_message(f"Criptomoeda: {self.crypto_var.get()}", "INFO")
        self.log_message(f"Período: {self.data_inicial_var.get()} até {self.
    data_final_var.get() or 'hoje'}", "INFO")

        thread = threading.Thread(target=self._execute_backtest)
        thread.daemon = True
        thread.start()
```

```
    def _execute_backtest(self):
        try:
            self.log_message("Preparando configurações...", "INFO")

            with open("instavel_att.py", "r", encoding="utf-8") as f:
                original_code = f.read()

            data_final = self.data_final_var.get().strip()
            if not data_final:
                data_final = "None"
            else:
                data_final = f"'{data_final}'"

            fee_value = self.fee_var.get()
            if fee_value > 1:
                fee_value = fee_value / 100.0
            slippage_value = self.slippage_var.get()
            if slippage_value > 1:
                slippage_value = slippage_value / 100.0

            modified_code = original_code
            modified_code = re.sub(r"^cripto = 'BTC-USD'", f"cripto = '{self.
    crypto_var.get()}'", modified_code, flags=re.MULTILINE)
            modified_code = re.sub(r"^data_inicial = '2021-01-01'", f"
    data_inicial = '{self.data_inicial_var.get()}'", modified_code, flags=re.
    MULTILINE)
            modified_code = re.sub(r"^data_final = '2022-01-01'.*?#", f"
    data_final = {data_final}  #", modified_code, flags=re.MULTILINE)
            modified_code = re.sub(r"^intervalo = '1h'", f"intervalo = '{self.
    interval_var.get()}'", modified_code, flags=re.MULTILINE)
            modified_code = re.sub(r"^taxa_operacional = 0.*?#", f"
    taxa_operacional = {fee_value}  #", modified_code, flags=re.MULTILINE)
            modified_code = re.sub(r"^derrapagem = 0.*?#", f"derrapagem = {
    slippage_value}    #", modified_code, flags=re.MULTILINE)
            modified_code = re.sub(r"^sma_media_rapida = 9", f"sma_media_rapida
     = {self.sma_fast_var.get()}", modified_code, flags=re.MULTILINE)
            modified_code = re.sub(r"^sma_media_lenta = 20", f"sma_media_lenta
    = {self.sma_slow_var.get()}", modified_code, flags=re.MULTILINE)
            modified_code = re.sub(r"^ema_media_rapida = 9", f"ema_media_rapida
```

Versão Final Homologada
27/12/2025 13:45

```
 = {self.ema_fast_var.get()}", modified_code, flags=re.MULTILINE)
        modified_code = re.sub(r"^ema_media_lenta = 20", f"ema_media_lenta
= {self.ema_slow_var.get()}", modified_code, flags=re.MULTILINE)
        modified_code = re.sub(r"^lsma_media_rapida = 10", f"
lsma_media_rapida = {self.lsma_fast_var.get()}", modified_code, flags=re.
MULTILINE)
        modified_code = re.sub(r"^lsma_media_lenta = 50", f"
lsma_media_lenta = {self.lsma_slow_var.get()}", modified_code, flags=re.
MULTILINE)

        temp_file = "temp_instavel_backtest.py"
        with open(temp_file, "w", encoding="utf-8") as f:
            f.write(modified_code)

        self.log_message("Arquivo de configuração criado", "SUCCESS")
        self.log_message("Executando backtest...", "INFO")

        result = subprocess.run(
            [sys.executable, temp_file],
            capture_output=True,
            text=True,
            encoding='utf-8',
            cwd=os.path.dirname(os.path.abspath("instavel_att.py"))
        )

        if result.stdout:
            for line in result.stdout.split('\n'):
                if line.strip():
                    self.log_message(line.strip(), "INFO")

        if result.returncode == 0:
            self.log_message("Backtest concluído com sucesso!", "SUCCESS")

            script_dir = os.path.dirname(os.path.abspath("instavel_att.py")
)
            bcktst_folders = [f for f in os.listdir(script_dir) if f.
startswith("bcktst_")]
            if bcktst_folders:
                bcktst_folders.sort(reverse=True)
                self.results_dir = os.path.join(script_dir, bcktst_folders
```

```
    [0])
                    self.log_message(f"Resultados salvos em: {self.results_dir
    }", "SUCCESS")
                    self.log_message("Arquivos gerados:", "INFO")

                    if os.path.exists(self.results_dir):
                        arquivos = os.listdir(self.results_dir)
                        for arquivo in arquivos:
                            if arquivo.endswith(('.html', '.csv', '.png', '.
    json')):
                                self.log_message(f"- {arquivo}", "INFO")

                    self.open_dashboard_button.config(state=tk.NORMAL)
                else:
                    self.log_message("Nenhuma pasta de resultados encontrada",
    "WARNING")
            else:
                self.log_message(f"Erro na execução: {result.stderr}", "ERROR")
                if result.stderr:
                    for line in result.stderr.split('\n'):
                        if line.strip():
                            self.log_message(line.strip(), "ERROR")

        except Exception as e:
            import traceback
            error_msg = traceback.format_exc()
            self.log_message(f"Erro inesperado: {str(e)}", "ERROR")
            self.log_message(f"Detalhes: {error_msg}", "ERROR")
        finally:
            temp_file = "temp_instavel_backtest.py"
            if os.path.exists(temp_file):
                try:
                    os.remove(temp_file)
                except:
                    pass

            self.is_running = False
            self.run_button.config(state=tk.NORMAL)
            self.stop_button.config(state=tk.DISABLED)
            self.status_var.set("Backtest concluído")
```

```
    def stop_backtest(self):
        self.is_running = False
        self.log_message("Parando backtest...", "WARNING")
        self.status_var.set("Backtest interrompido")

    def open_dashboard(self):
        if not self.results_dir or not os.path.exists(self.results_dir):
            messagebox.showerror("Erro", "Nenhum resultado encontrado. Execute
    o backtest primeiro.")
            return

        dashboard_path = os.path.join(self.results_dir, "dashboard_unificado.
    html")
        if os.path.exists(dashboard_path):
            webbrowser.open(f"file:///{os.path.abspath(dashboard_path)}")
            self.log_message("Dashboard aberto no navegador", "SUCCESS")
        else:
            messagebox.showerror("Erro", f"Dashboard não encontrado em: {
    dashboard_path}")

    def open_folder(self):
        if self.results_dir and os.path.exists(self.results_dir):
            folder_path = self.results_dir
        else:
            folder_path = os.path.dirname(os.path.abspath(__file__))

        if os.name == 'nt':
            os.startfile(folder_path)
        elif os.name == 'posix':
            subprocess.run(['open' if sys.platform == 'darwin' else 'xdg-open',
     folder_path])
        self.log_message("Pasta aberta", "INFO")

def main():
    root = tk.Tk()
    app = InstavelBacktestGUI(root)

    def on_closing():
        if app.is_running:
```

```
            if messagebox.askokcancel("Sair", "Backtest em execução. Deseja
    realmente sair?"):
                root.destroy()
        else:
            root.destroy()

    root.protocol("WM_DELETE_WINDOW", on_closing)
    root.mainloop()

if __name__ == "__main__":
    main()
```