# Guia Mestre de Implementação: Bot de Criptomoedas para Telegram

## Seção 1: Configuração Fundamental e Esboço Arquitetural

Esta seção estabelece a base do projeto, detalhando sua visão, objetivos e a arquitetura técnica que guiará a implementação. Ela traduz os requisitos de alto nível, conforme delineado no Documento de Requisitos do Produto (PDR2.pdf), e a estrutura visual do mapa-mental-bot-cripto.md em um plano arquitetural tangível, preparando o terreno para as etapas práticas de desenvolvimento.1

### 1.1 Visão e Objetivos do Projeto

O objetivo central deste projeto é desenvolver um bot pessoal para o Telegram, projetado para o monitoramento de criptomoedas. As funcionalidades essenciais incluem a consulta de preços em tempo real, a criação de alertas de preço personalizados, a obtenção de análises técnicas e o acesso a notícias relevantes do mercado. O projeto é concebido para ser uma ferramenta de estudo e uso pessoal, com foco em automação utilizando a plataforma N8N.1

O perfil do usuário final, um estudante de automação de 53 anos em transição de carreira, é um fator determinante que molda a abordagem deste guia. O foco será na clareza, no valor educacional e na aplicação de padrões de desenvolvimento robustos, em vez de complexidade de nível empresarial.1

O sucesso do projeto será medido com base em métricas de desempenho claras, que servirão como um guia para a garantia de qualidade ao longo do desenvolvimento 1:

* **Tempo de Resposta:** As interações do usuário devem ser processadas em menos de 3 segundos.
* **Latência de Alerta:** Os alertas de preço devem ser disparados em menos de 60 segundos após a condição ser atendida.
* **Uptime:** O sistema deve manter uma disponibilidade de 99%.
* **Precisão:** O sistema deve operar com zero falsos alertas.

### 1.2 Estrutura Conceitual: Integrando o Mapa Mental

Este guia foi estruturado para ser a representação textual e operacional do mapa-mental-bot-cripto.md. A organização das seções reflete diretamente os componentes principais e suas interconexões, como detalhado no mapa mental.

* **O Conceito de Mapa Mental:** A estrutura do guia, especialmente a forma como detalha a interação entre os componentes, funciona como um mapa mental conceitual. As seções ilustram como os workflows do N8N (o núcleo de automação), os modelos do banco de dados Supabase (a camada de persistência), as integrações de API (as fontes de dados externas) e o Agente LangChain (o cérebro de IA) se interconectam para formar um sistema coeso.

### 1.3 Pilha de Tecnologia e Arquitetura do Sistema

A seleção de tecnologias, conforme especificado no documento de requisitos, forma uma pilha moderna e eficiente para automação e manipulação de dados.1

* **Automação:** N8N v1.107.3 (hospedado via Hostinger/Portainer)
* **Banco de Dados:** Supabase (utilizando PostgreSQL)
* **APIs Externas:** Binance (para dados de preço), TAAPI.io (para análise técnica), CryptoCompare (para notícias) e OpenAI (para processamento de linguagem natural).
* **Interface com o Usuário:** Telegram Bot

A arquitetura do sistema é sustentada por três workflows principais no N8N, cada um com uma responsabilidade distinta, promovendo a separação de preocupações e a manutenibilidade 1:

1. WF\_Bot\_Principal: Gerencia todas as interações diretas com o usuário.
2. WF\_Monitor\_Alertas: Opera em segundo plano, verificando continuamente as condições dos alertas de preço.
3. WF\_Error\_Handler: Um workflow centralizado para capturar, registrar e notificar sobre erros, garantindo a resiliência do sistema.

O sistema também emprega três agentes de IA especializados, construídos com a flexibilidade do LangChain, para tarefas de processamento inteligente 1:

1. **Agente Orquestrador (LangChain):** Atua como o cérebro principal, interpretando a linguagem natural do usuário para não apenas extrair intenções e entidades, mas também para orquestrar a sequência de ferramentas necessárias para cumprir a solicitação.
2. **Analista:** Resume dados técnicos brutos em uma análise compreensível.
3. **Curador:** Filtra e resume notícias para apresentar as mais relevantes.

### 1.4 Diagrama de Fluxo de Dados (DFD)

O documento dfd-bot-cripto.md detalha o percurso que os dados fazem através do sistema. A seguir, uma descrição textual desse fluxo para uma solicitação típica, como "preço do btc":

1. **Entrada do Usuário:** O usuário envia a mensagem "preço do btc" para o bot no Telegram.
2. **Gatilho N8N:** O Telegram envia um evento para a URL do webhook configurada no nó **Telegram Trigger** do WF\_Bot\_Principal.
3. **Orquestração de IA:** A mensagem é passada para o **Agente LangChain**. O agente interpreta a intenção (get\_price) e a entidade (BTC).
4. **Roteamento:** O WF\_Bot\_Principal roteia a execução para a lógica da ferramenta get\_price.
5. **Verificação de Cache:** O workflow primeiro consulta a tabela PriceCache no **Supabase** para verificar se há um preço válido (com menos de 30 segundos).
6. **Chamada de API (se necessário):** Se o cache estiver expirado ou não existir, o workflow faz uma chamada GET para a API da **Binance**.
7. **Persistência de Cache:** O novo preço obtido da Binance é salvo na tabela PriceCache do **Supabase**.
8. **Formatação da Resposta:** O workflow formata os dados do preço em uma mensagem amigável.
9. **Saída para o Usuário:** A mensagem formatada é enviada de volta ao usuário via **Telegram**.

Este fluxo demonstra a interação orquestrada entre a interface do usuário, a lógica de automação, a camada de IA, o banco de dados e os serviços externos.

### 1.5 Casos de Uso Principais

Para garantir que a implementação atenda diretamente às necessidades do usuário, os seguintes casos de uso, derivados dos requisitos do projeto, servem como guia para o desenvolvimento das funcionalidades.1

**Tabela 1.1: Mapeamento de Casos de Uso para Funcionalidades**

| ID do Caso de Uso | Descrição do Caso de Uso | Funcionalidade Correspondente |
| --- | --- | --- |
| CU-01 | Como usuário, quero consultar o preço atual e as estatísticas de 24h de uma criptomoeda para tomar decisões de investimento. | Preço |
| CU-02 | Como usuário, quero criar um alerta para ser notificado quando uma criptomoeda atingir um determinado preço (acima ou abaixo). | Criar Alerta |
| CU-03 | Como usuário, quero ver uma lista de todos os meus alertas de preço ativos para poder gerenciá-los. | Meus Alertas |
| CU-04 | Como usuário, quero remover um alerta que não é mais necessário. | Meus Alertas (Remoção) |
| CU-05 | Como usuário, quero obter as notícias mais recentes e o sentimento do mercado para uma criptomoeda específica. | Notícias |
| CU-06 | Como usuário, quero receber uma análise técnica simplificada (RSI, MACD, etc.) para avaliar a tendência de um ativo. | Análise Técnica |
| CU-07 | Como novo usuário, quero receber uma mensagem de boas-vindas e instruções claras sobre como usar o bot. | /start, /help |

## Seção 2: Implementação do Banco de Dados: Do Esquema à Instância Supabase Ativa

Esta seção oferece um guia prático para a criação da camada de persistência de dados do projeto. O processo envolve a tradução do modelo de dados abstrato, definido no arquivo prisma-schema-bot.txt, para uma instância Supabase totalmente configurada e pronta para se conectar ao N8N.1

### 2.1 Configurando seu Projeto Supabase

A primeira etapa é criar e configurar o ambiente do banco de dados na plataforma Supabase.

1. **Criação do Projeto:** Crie uma nova organização e um novo projeto em sua conta Supabase. Durante a criação do projeto, é fundamental definir uma senha segura para o banco de dados e salvá-la em um local seguro, pois ela será necessária posteriormente para a configuração da conexão no N8N.3
2. **Habilitação de Extensões (Opcional, mas Recomendado):** Para futuras aplicações de IA, como a busca semântica, é aconselhável habilitar a extensão pgvector. Isso pode ser feito executando o comando SQL apropriado no Editor SQL do Supabase.3
3. **Coleta de Credenciais:** Após a criação do projeto, navegue até as configurações do projeto para coletar as chaves de API e os detalhes de conexão do banco de dados, que serão utilizados na próxima etapa.4

### 2.2 A "Configuração de Nó Duplo": Um Padrão Crítico de Conexão N8N-Supabase

Uma particularidade importante e não óbvia da integração entre N8N e Supabase é a necessidade de uma "configuração de nó duplo". Para interagir plenamente com o Supabase, são necessários dois conjuntos de credenciais e dois tipos de nós diferentes no N8N.4

* **Justificativa:** A necessidade dessa abordagem dupla decorre das duas maneiras distintas de interagir com o Supabase:
  1. **Via API:** O **nó Supabase** no N8N interage com a API REST do Supabase. Ele é ideal para operações CRUD (Criar, Ler, Atualizar, Deletar) simples e diretas em linhas de tabelas.5
  2. **Via Conexão Direta ao Banco de Dados:** O **nó Postgres** no N8N estabelece uma conexão direta com o banco de dados PostgreSQL subjacente do Supabase. Esta conexão é essencial para executar consultas SQL complexas, interagir com funções do banco de dados e conectar-se a nós especializados do N8N, como o Supabase Vector Store ou o Postgres Chat Memory.4
* **Guia de Configuração:**
  1. **Credenciais da API Supabase:** No N8N, crie uma nova credencial do tipo "Supabase API". Preencha o campo "Host" com a URL do seu projeto Supabase (encontrada em Project Settings > API) e o campo "Service Role Secret" com a chave service\_role correspondente.4
  2. **Credenciais do Postgres:** Crie uma segunda credencial no N8N, desta vez do tipo "Postgres". Utilize os detalhes de conexão do "pooler de transações" (Transaction pooler) encontrados no painel do Supabase. Isso inclui o Host, a Porta (geralmente 6543 para o pooler), o Banco de Dados (postgres), o Usuário (postgres) e a senha do banco de dados que foi definida durante a criação do projeto. É importante definir a opção SSL como "desabilitada" (disable) para esta conexão específica.4

### 2.3 Implantando o Esquema Prisma: Uma Análise Detalhada Modelo a Modelo

O arquivo prisma-schema-bot.txt define a estrutura completa e robusta do banco de dados. A seguir, uma análise do propósito de cada modelo de dados 1:

* BotConfig: Armazena configurações globais do bot, permitindo ajustes dinâmicos sem a necessidade de alterar os workflows.
* UserSettings: Projetado para armazenar informações e preferências do usuário. Embora o projeto seja para um único usuário, esta tabela já prepara a arquitetura para um futuro suporte a múltiplos usuários.
* Alert: Tabela central para a funcionalidade de alertas de preço. Armazena o símbolo da criptomoeda, a condição (above ou below), o preço-alvo e o estado do alerta (isActive).
* PriceCache, TechnicalAnalysis, NewsCache: Estas tabelas implementam a estratégia de cache definida nas regras de negócio do PRD.1 Elas são cruciais para otimizar o desempenho, reduzir a latência e gerenciar os custos e limites das APIs externas.
* BotLog, ApiUsage, ErrorState: Um conjunto de tabelas dedicadas ao registro de logs, monitoramento do uso de APIs e rastreamento de estados de erro. A presença dessas tabelas demonstra uma mentalidade orientada para a produção, focada em observabilidade e depuração.

A estrutura do banco de dados, conforme definida no esquema Prisma, é notavelmente mais detalhada do que a simples lista de tabelas sugerida no PRD. O esquema inclui índices otimizados para consultas frequentes, relacionamentos de chave estrangeira bem definidos e tipos de dados precisos (como Decimal(20, 8)), ideais para dados financeiros.1 Uma das decisões de design mais significativas é a inclusão de um modelo

UserSettings completo e a vinculação de tabelas como Alert e BotLog a ele através de um userId.

Esta abordagem prepara a arquitetura do banco de dados para o suporte a múltiplos usuários desde o início, um item listado como uma "Consideração Futura" no PRD.1 A implementação dessa funcionalidade futura exigirá apenas modificações na lógica da aplicação no N8N para gerenciar diferentes

userIds, em vez de uma migração de banco de dados complexa e arriscada. Este é um exemplo claro de design previdente, que não apenas facilita a evolução do projeto, mas também ensina um princípio valioso de engenharia de software.

**Tabela 2.1: Mapeamento de Modelos de Banco de Dados para Funcionalidades**

| Modelo Prisma 1 | Função Principal | Funcionalidade do PRD Suportada 1 | Campos-Chave |
| --- | --- | --- | --- |
| Alert | Armazena alvos de preço e condições definidos pelo usuário. | Criar Alerta, Meus Alertas | userId, symbol, condition, targetPrice, isActive |
| PriceCache | Armazena temporariamente os preços das criptomoedas para reduzir chamadas de API. | Preço | symbol, price, changePercent24h, updatedAt |
| TechnicalAnalysis | Armazena em cache os resultados da análise técnica. | Análise Técnica | symbol, timeframe, rsi, macdInterpretation, expiresAt |
| NewsCache | Armazena em cache as notícias e o sentimento do mercado. | Notícias | title, url, relatedTokens, sentiment |
| UserSettings | Armazena dados e preferências do usuário. | (Suporte futuro para múltiplos usuários) | userId, username, languageCode, isActive |
| BotLog | Registra eventos e operações do sistema para depuração. | (Operações internas, WF\_Error\_Handler) | workflowName, action, errorMessage, createdAt |

## Seção 3: Implementação dos Workflows Principais (Fases 1-2)

Esta seção inicia o desenvolvimento prático no N8N, focando na construção da espinha dorsal do bot: o workflow principal de interação e o manipulador de erros global. Estas são as fases 1 e 2 do plano de implementação delineado no PRD.1

### 3.1 Construindo o WF\_Error\_Handler: Uma Rede de Segurança Reutilizável

A criação de um workflow dedicado e reutilizável para o tratamento de erros é uma prática recomendada que aumenta a robustez e a manutenibilidade do sistema, conforme especificado na arquitetura.1

* **Conceito:** Este workflow centralizará toda a lógica de tratamento de falhas. Em vez de replicar a lógica de log e notificação em cada workflow, outros workflows simplesmente chamarão o WF\_Error\_Handler quando um erro ocorrer.
* **Passos de Implementação:**
  1. Crie um novo workflow no N8N e nomeie-o WF\_Error\_Handler.
  2. Utilize um nó **Webhook** como gatilho. Isso permite que o workflow seja invocado por meio de uma chamada HTTP, passando os dados do erro (como a mensagem de erro, o nome do workflow e o nó onde ocorreu a falha) no corpo da requisição.
  3. Adicione um nó **Supabase** configurado no modo "Insert". Mapeie os dados de erro recebidos pelo webhook para os campos correspondentes na tabela BotLog ou ErrorState do banco de dados.1
  4. Adicione um nó **Telegram** para enviar uma mensagem de notificação para o administrador (o ID de chat do usuário) contendo os detalhes essenciais do erro.
  5. Para invocar este manipulador, utilize o nó **Execute Workflow** dentro de outros workflows. Em cenários de falha, o nó **Error Trigger** (que funciona como um bloco catch) pode ser conectado ao nó **Execute Workflow** para passar os detalhes do erro para o WF\_Error\_Handler.

### 3.2 Construindo o WF\_Bot\_Principal: O Cérebro de Interação Central

Este workflow é o ponto de entrada para todas as interações do usuário e orquestra a resposta do bot.

* **Gatilho:** Configure o nó **Telegram Trigger** utilizando o token da API do seu bot do Telegram. Este nó escutará por novas mensagens enviadas ao bot.
* **Comandos Iniciais (/start, /help):**
  1. Conecte o gatilho a um nó **Switch**. Este nó atuará como um roteador inicial, verificando o conteúdo da mensagem.
  2. Crie uma saída no nó **Switch** para a condição em que a mensagem é igual a /start. Conecte esta saída a um nó **Telegram** que enviará a mensagem de boas-vindas e o menu com botões inline (teclado), conforme definido no PRD.1
  3. Crie outra saída para a condição /help, conectando-a a um nó **Telegram** que lista os comandos disponíveis e instruções de uso.
* **Implementando o Cérebro do Bot: O Agente LangChain:**
  1. Para todas as outras mensagens (a saída padrão do nó **Switch**), a lógica de interpretação de linguagem natural será acionada. Conecte esta saída a um nó **AI Agent**.
  2. **Configuração:**
     + No nó **AI Agent**, defina o tipo de agente como **Tools Agent**.7
     + Conecte um nó de modelo de linguagem, como o **OpenAI Chat Model** ou **Google Gemini Chat Model**, à entrada "Chat Model" do agente.7
     + **Mensagem de Sistema (System Message):** Este é o passo mais crítico. Elabore um prompt que instrua o modelo de IA a atuar como um "Agente Orquestrador". O prompt deve orientá-lo a analisar a mensagem do usuário (ex: "BTC", "preço do bitcoin", "criar alerta para ethereum abaixo de 3000") e, em vez de apenas extrair dados, determinar qual ferramenta ou sequência de ferramentas deve ser usada para atender à solicitação. O objetivo é que o agente decida a ação a ser tomada (a "intenção") e colete os parâmetros necessários.7 A saída deve ser uma chamada de ferramenta estruturada. Por exemplo, para a entrada "preço do bitcoin", o agente deve decidir chamar a ferramenta  
       get\_price com o parâmetro symbol: "BTC".
* **Roteador de Comandos:**
  1. Conecte a saída do nó **AI Agent** a um segundo nó **Switch**.
  2. Configure este nó para rotear o fluxo com base no nome da ferramenta que o agente LangChain decidiu usar. Cada saída deste switch corresponderá a uma funcionalidade do bot (ex: get\_price, create\_alert, get\_news), direcionando para a lógica específica que será implementada na Seção 4.

A decisão de usar um **Agente LangChain** como o cérebro central do bot é uma escolha arquitetural fundamental que eleva a capacidade do sistema.1 Uma abordagem tradicional de "parsing" com expressões regulares ou nós "If" é frágil e de difícil manutenção. Ao delegar essa tarefa a um LLM através do nó

**AI Agent** do N8N, que utiliza LangChain internamente, o bot ganha uma flexibilidade imensa.8 Ele se torna capaz de compreender uma vasta gama de variações de linguagem natural ("qual o preço do bitcoin?", "quanto está valendo o btc?", "btc cotação") sem que cada possibilidade precise ser codificada explicitamente. Mais importante, o agente pode raciocinar sobre qual ferramenta usar, tornando o bot mais inteligente e adaptável.2 Isso simplifica o restante do workflow, que pode operar com base nas decisões estruturadas do agente, em vez de lidar com a complexidade do texto livre.

## Seção 4: Implementação de Funcionalidades Avançadas e Integração de APIs (Fases 3-4)

Nesta seção, são construídas as funcionalidades centrais que entregam valor ao usuário. O foco está na integração com APIs externas e na implementação dos workflows restantes e dos agentes de IA especializados, correspondendo às fases 3 e 4 do plano de implementação.1

### 4.1 Construindo o WF\_Monitor\_Alertas: O Vigia de Preços em Segundo Plano

Este workflow autônomo é responsável por monitorar os alertas de preço de forma contínua e proativa.

* **Gatilho:** Utilize um nó **Schedule** (Agendador) configurado para ser executado a cada 1 minuto. Esta frequência está alinhada com a regra de negócio de verificação de 60 segundos definida no PRD.1
* **Lógica de Execução:**
  1. **Consulta ao Banco de Dados:** Adicione um nó **Postgres** (utilizando as credenciais de conexão direta) para executar uma consulta SQL que seleciona todos os alertas ativos: SELECT \* FROM alerts WHERE is\_active = true;.1
  2. **Processamento em Lote:** Conecte a saída do nó Postgres a um nó **Loop Over Items**. Este nó irá iterar sobre cada alerta ativo retornado pela consulta, processando um de cada vez.
  3. **Busca de Preço Atual:** Dentro do loop, adicione um nó **HTTP Request** para fazer uma chamada à API da Binance e obter o preço atual do símbolo (symbol) do alerta em processamento.
  4. **Verificação da Condição:** Utilize um nó **If** para comparar o preço atual com o targetPrice do alerta, respeitando a condition (above ou below). A condição seria algo como: {{ $json.current\_price }} > {{ $json.targetPrice }}.
  5. **Ação de Disparo:** Se a condição do nó **If** for verdadeira:
     + Conecte a saída "true" a um nó **Telegram** para enviar a mensagem de notificação ao userId associado ao alerta.
     + Em seguida, adicione um nó **Supabase** configurado no modo "Update". Este nó deve atualizar o registro do alerta no banco de dados, definindo is\_active = false e notification\_sent = true. Esta etapa é crucial para garantir que o alerta seja desativado após o disparo, evitando notificações repetidas, conforme a regra de negócio.[1, 1]

### 4.2 Implementando Funcionalidades Sob Demanda (dentro do WF\_Bot\_Principal)

Esta seção expande o roteador de comandos construído na Seção 3.2, implementando a lógica para cada ferramenta que o Agente LangChain pode decidir usar.

* **Ferramenta de Preço (tool: "get\_price")**
  1. **Lógica de Cache (Leitura):** Antes de chamar a API, adicione um nó **Supabase** (modo "Get Many") para verificar se existe um preço recente para o símbolo na tabela price\_cache. A consulta deve filtrar por registros onde updatedAt seja nos últimos 30 segundos.1 Um nó  
     **If** subsequente verificará se a consulta retornou algum resultado.
  2. **Chamada de API:** Se não houver cache válido, execute a chamada de API. Utilize um nó **HTTP Request** para consultar o endpoint de estatísticas de 24 horas da Binance: GET /fapi/v1/ticker/24hr, passando o símbolo da criptomoeda como um parâmetro de consulta (symbol).
  3. **Extração de Dados:** Processe a resposta JSON da API para extrair os campos lastPrice, priceChangePercent e volume.
  4. **Lógica de Cache (Escrita):** Após uma chamada de API bem-sucedida, use um nó **Supabase** (modo "Update" ou "Insert") para salvar ou atualizar o resultado na tabela price\_cache, garantindo que o updatedAt seja registrado.1
  5. **Formatação da Resposta:** Unifique os fluxos (de cache e de API) e formate os dados na mensagem clara e concisa especificada no PRD, antes de enviá-la ao usuário com um nó **Telegram**.1
* **Ferramenta de Análise Técnica (tool: "get\_analysis")**
  1. **Lógica de Cache (Leitura):** Implemente a verificação de cache na tabela technical\_analysis, buscando por uma análise para o símbolo e timeframe nos últimos 15 minutos.1
  2. **Chamada de API:** Se o cache estiver expirado, utilize o nó **HTTP Request** para chamar a API da **TAAPI.io**. Serão necessárias chamadas aos endpoints específicos para os indicadores RSI, StochRSI e MACD.
  3. **Agente de IA "Analista":** Os valores brutos dos indicadores (ex: RSI: 68, MACD: Bullish) devem ser passados como entrada para um segundo nó **AI Agent**. A mensagem de sistema para este agente "Analista" deve ser: "Com base nos seguintes indicadores técnicos para [símbolo], forneça um resumo conciso e em linguagem natural sobre a tendência atual do mercado (ex: 'A tendência é de alta, com forte momento de compra').".1
  4. **Formatação e Cache (Escrita):** Combine os dados brutos dos indicadores com o resumo gerado pela IA no formato de mensagem final. Salve este resultado compilado na tabela technical\_analysis antes de enviá-lo ao usuário.1
* **Ferramenta de Notícias (tool: "get\_news")**
  1. **Lógica de Cache (Leitura):** Verifique a tabela news\_cache por notícias relacionadas ao token nos últimos 30 minutos.1
  2. **Chamada de API:** Se necessário, utilize o nó **HTTP Request** para consultar o endpoint de notícias da API da **CryptoCompare**: GET /data/v2/news/. É possível filtrar as notícias por categoria ou relevância para o símbolo em questão.
  3. **Agente de IA "Curador":** A lista de manchetes e resumos das notícias obtidas da API deve ser enviada a um terceiro nó **AI Agent**. A mensagem de sistema para este agente "Curador" será: "Desta lista de notícias, selecione as 2 manchetes mais impactantes para um investidor. Em seguida, determine o sentimento geral do mercado (BULLISH, BEARISH ou NEUTRAL) com base nessas notícias.".1
  4. **Formatação e Cache (Escrita):** Formate as manchetes selecionadas e o sentimento determinado pela IA na mensagem final. Armazene este resultado curado na tabela news\_cache antes de enviar ao usuário.1

A estratégia de cache, explicitamente definida no PRD e suportada por tabelas dedicadas no esquema do banco de dados, é um pilar central da arquitetura do sistema.[1, 1] Em um sistema que depende de múltiplas APIs de terceiros, algumas com planos gratuitos e limites de taxa restritos (como a TAAPI.io ), o cache deixa de ser uma mera otimização de desempenho e se torna um componente essencial para a viabilidade operacional e econômica. Ele garante uma experiência de usuário mais rápida, pois a recuperação de dados de um banco de dados local é significativamente mais veloz do que uma chamada de API externa, ao mesmo tempo que respeita os limites de uso e minimiza os custos. Portanto, a implementação da lógica de cache não é opcional, mas sim um passo mandatório em cada fluxo que envolve dados externos, garantindo a sustentabilidade e a eficiência do bot.

## Seção 5: Runbook Operacional: Testes, Implantação e Monitoramento (Fase 5)

Esta seção finaliza a transição do projeto do ambiente de desenvolvimento para um estado operacional e ativo. Ela serve como o "runbook" do projeto, fundindo o checklist-deploy-bot e o runbook-bot-cripto.md em um guia prático para garantir a qualidade, segurança e confiabilidade do bot, conforme a fase 5 do plano de implementação.1

### 5.1 Testes e Garantia de Qualidade

A fase de testes é crucial para validar que o sistema atende a todos os requisitos funcionais e de desempenho. Uma lista de verificação baseada diretamente nas "Métricas de Sucesso" do PRD deve ser utilizada.1

* **Lista de Verificação de Testes Técnicos:**
  + **Tempo de Resposta (< 3s):** Em cada fluxo do WF\_Bot\_Principal, utilize nós de data/hora no início e no fim para calcular o tempo de execução e verificar se está dentro do limite.
  + **Latência de Alerta (< 60s):** Crie um alerta de teste com um preço-alvo muito próximo do preço atual (ex: 1% acima). Meça o tempo entre a criação do alerta e o recebimento da notificação para garantir que o WF\_Monitor\_Alertas está operando dentro da latência esperada.
  + **Uptime (99%):** A disponibilidade do bot depende da disponibilidade do N8N e do webhook. A monitorização pode ser configurada usando um serviço externo como UptimeRobot, que pode verificar periodicamente a URL do webhook do WF\_Bot\_Principal para garantir que está acessível.
  + **Zero Falsos Alertas:** Execute um teste de longa duração (soak test) com múltiplos alertas configurados (acima e abaixo do preço atual) para verificar que nenhum alerta é disparado incorretamente.
* **Testes de Aceitação do Usuário (UAT):**
  + O usuário final deve interagir com o bot e testar cada comando e funcionalidade.
  + Verificar se todas as mensagens de boas-vindas, ajuda, erro, e os resultados de preço, análise e notícias correspondem exatamente aos formatos especificados no PRD.1
  + Testar a criação e remoção de alertas, bem como a clareza das notificações.

### 5.2 Procedimentos de Implantação (Runbook)

Esta seção detalha os passos operacionais para a implantação, combinando as melhores práticas de segurança com um checklist prático.

* **Melhores Práticas de Segurança:**
  1. **Gerenciamento de Chaves de API:** É de importância crítica que todas as chaves de API e segredos (Telegram, Binance, OpenAI, TAAPI.io, CryptoCompare, Supabase) sejam armazenados exclusivamente no **N8N Vault**. Elas nunca devem ser inseridas diretamente nos campos dos nós. Esta prática, especificada no PRD, evita a exposição de credenciais sensíveis no código do workflow.1
  2. **Segurança do Webhook:** Para proteger o webhook do Telegram contra chamadas não autorizadas, um token secreto pode ser adicionado à URL do webhook. Esta funcionalidade, se suportada pelo nó do Telegram, adiciona uma camada de verificação.
  3. **Sanitização de Entradas:** Embora o uso do Agente LangChain para estruturar a entrada do usuário já mitigue muitos riscos, é uma boa prática garantir que os dados de entrada sejam validados sempre que possível antes de serem usados em consultas de banco de dados ou chamadas de API.1
* **Checklist de Implantação:**
  1. **Configuração de Ambiente:**
     + [ ] Instância N8N (Hostinger/Portainer) configurada e acessível.
     + [ ] Projeto Supabase criado e esquema de banco de dados implantado.
  2. **Credenciais e Segurança:**
     + [ ] Todas as chaves de API (Telegram, Binance, etc.) armazenadas de forma segura no N8N Vault.1
     + [ ] Credenciais de "nó duplo" (API Supabase e Postgres) configuradas e testadas.4
     + [ ] Webhook do Telegram protegido com token secreto.1
  3. **Ativação dos Workflows (Ordem de Ativação):**
     + [ ] Ativar WF\_Error\_Handler e testar sua invocação.
     + [ ] Ativar WF\_Monitor\_Alertas e verificar se o agendamento está ativo.
     + [ ] Ativar WF\_Bot\_Principal e registrar o webhook no Telegram.
  4. **Testes Finais (Go/No-Go):**
     + [ ] Execução de todos os casos de teste da Seção 5.1 com sucesso.
     + [ ] Verificação do log de erros no Supabase (tabela bot\_logs) para garantir que não há falhas na inicialização.1
  5. **Monitoramento Pós-Implantação:**
     + [ ] Configuração de monitoramento de uptime (ex: UptimeRobot) para o webhook.
     + [ ] Verificação manual das primeiras interações do usuário e execuções do monitor de alertas.
     + [ ] Revisão periódica da tabela bot\_logs e api\_usage no Supabase para monitorar a saúde do sistema e o consumo de API.

### 5.3 Gerenciamento de Riscos e Mitigação

A matriz-riscos-bot.md identifica potenciais pontos de falha e estratégias para mitigá-los. A arquitetura atual já endereça os principais riscos:

* **Risco: Falha ou Indisponibilidade de APIs Externas (Binance, TAAPI.io, etc.)**
  + **Mitigação:** A estratégia de cache implementada (tabelas PriceCache, TechnicalAnalysis, NewsCache) reduz a dependência de chamadas constantes. Em caso de falha, o WF\_Error\_Handler é acionado, registrando o erro no Supabase e notificando o administrador imediatamente para uma ação rápida.
* **Risco: Limites de Taxa de API (Rate Limiting), especialmente no plano gratuito da TAAPI.io**
  + **Mitigação:** O cache de 15 minutos para análises técnicas diminui drasticamente o número de chamadas. A tabela ApiUsage permite monitorar o consumo e identificar se os limites estão sendo aproximados, possibilitando ajustes proativos.
* **Risco: Erros de Interpretação do Agente de IA**
  + **Mitigação:** O prompt de sistema do Agente LangChain foi projetado para ser claro e diretivo. A tabela ErrorState no Supabase foi criada especificamente para registrar entradas que a IA não conseguiu processar, permitindo uma análise posterior e o refinamento contínuo dos prompts.
* **Risco: Perda de Dados ou Corrupção do Banco de Dados**
  + **Mitigação:** O Supabase oferece backups automáticos como parte de sua infraestrutura, garantindo a capacidade de restauração em caso de um evento catastrófico.

### 5.4 Considerações Futuras e Evolução do Projeto

Uma revisão das melhorias futuras listadas no PRD ajuda a contextualizar o valor da arquitetura atual e a planejar os próximos passos.1

* **Melhorias Futuras:** Dashboard web, mais indicadores técnicos, exportação de dados, suporte a múltiplos usuários e backtesting de estratégias.
* **Prontidão Arquitetural:** É importante destacar como as decisões de design tomadas no início facilitam essas futuras expansões. Por exemplo, como discutido na Seção 2, o banco de dados já está estruturado para suportar múltiplos usuários através do userId. Habilitar essa funcionalidade agora é uma tarefa de lógica de aplicação no N8N, e não uma reestruturação complexa do banco de dados. Esta conexão entre o design atual e os objetivos futuros reforça o valor do trabalho realizado e demonstra uma abordagem de desenvolvimento madura e previdente.

## Apêndice A: Referência de Projeto e APIs

Este apêndice serve como um registro formal dos documentos que fundamentaram este guia e como uma referência rápida para as APIs externas utilizadas.

### A.1 Documentos de Projeto Analisados

* PDR2.pdf 1:  
  O Documento de Requisitos do Produto, definindo a visão, funcionalidades e regras de negócio.
* prisma-schema-bot.txt 1:  
  O esquema detalhado do banco de dados para o Supabase.
* **mapa-mental-bot-cripto.md:** A estrutura visual e conceitual dos componentes do sistema.
* **dfd-bot-cripto.md:** O detalhamento do fluxo de dados através do sistema.
* **runbook-bot-cripto.md:** Os procedimentos operacionais para implantação e manutenção.
* **matriz-riscos-bot.md:** A identificação de riscos do projeto e suas estratégias de mitigação.
* **Outros Documentos:** As informações conceituais dos documentos casos-uso-bot-cripto, apis-integracoes-bot e checklist-deploy-bot foram integradas diretamente nas seções relevantes deste guia.

### A.2 Tabela de Referência Rápida de Endpoints de API Externos

| Serviço | Endpoint | Propósito | Parâmetros-Chave | Referência |
| --- | --- | --- | --- | --- |
| **Binance** | GET /fapi/v1/ticker/24hr | Obter estatísticas de preço das últimas 24h (último preço, % de mudança, volume). | symbol |  |
| **TAAPI.io** | GET /rsi GET /stochrsi GET /macd | Obter valores de indicadores de análise técnica. | secret, exchange, symbol, interval |  |
| **CryptoCompare** | GET /data/v2/news/ | Obter os artigos de notícias mais recentes. | lang, api\_key |  |

## Apêndice B: Documentos Legais e de Conformidade

Esta seção referencia os documentos que governam o uso do bot e o tratamento dos dados do usuário, garantindo transparência e conformidade.

### B.1 Política de Privacidade (politica-privacidade-bot.md)

Este documento detalha como os dados do usuário (como o ID do Telegram e os alertas criados) são coletados, armazenados e processados. Ele assegura ao usuário que suas informações são utilizadas exclusivamente para o funcionamento do bot e não são compartilhadas com terceiros.

### B.2 Termos de Serviço (termos-servico-bot.md)

Este documento estabelece as regras e diretrizes para o uso do bot. Ele informa ao usuário sobre suas responsabilidades, as limitações do serviço (por exemplo, não se trata de aconselhamento financeiro) e as condições sob as quais o serviço é fornecido.

#### Trabalhos citados

1. PDR2.pdf
2. LangChain concepts in n8n | n8n Docs, acesso a agosto 30, 2025, <https://docs.n8n.io/advanced-ai/langchain/langchain-n8n/>
3. How to connect Supabase and Postgres to n8n - Optimize Smart, acesso a agosto 30, 2025, <https://www.optimizesmart.com/how-to-connect-supabase-and-postgres-to-n8n/>
4. How to Connect Supabase to n8n: The Dual Node Setup Guide, acesso a agosto 30, 2025, <https://blog.horizon.dev/connect-supabase-to-n8n/>
5. Postgres and Supabase: Automate Workflows with n8n, acesso a agosto 30, 2025, <https://n8n.io/integrations/postgres/and/supabase/>
6. How to Set Up Supabase and Postgres for Memory and RAG in n8n - YouTube, acesso a agosto 30, 2025, <https://www.youtube.com/watch?v=6w5f_jsPYSQ>
7. How To Build Your First AI Agent (+Free Workflow Template) – n8n ..., acesso a agosto 30, 2025, <https://blog.n8n.io/how-to-build-ai-agent/>
8. n8n vs. LangChain: A Comparative Guide for AI-Driven Workflow Automation | by Shiv, acesso a agosto 30, 2025, <https://medium.com/@shiv0307/n8n-vs-langchain-a-comparative-guide-for-ai-driven-workflow-automation-19cc20e13465>
9. Langchain vs n8n - Reddit, acesso a agosto 30, 2025, <https://www.reddit.com/r/LangChain/comments/1ige7m2/langchain_vs_n8n/>