Diagramas Mermaid – Bot Cripto Telegram

# Diagrama 1 – Arquitetura Geral

```mermaid  
flowchart TD  
 Telegram[Telegram Bot] -->|Mensagens| N8N[WF\_Bot\_Principal]  
 N8N --> LangChain[Agente IA (LangChain + MCP)]  
 LangChain --> Redis[(Redis - Cache)]  
 LangChain --> Supabase[(Supabase/Postgres)]  
 LangChain --> Binance[Binance Spot API]  
 LangChain --> RSS[RSS Feeds Notícias]  
```

# Diagrama 2 – Fluxo do Comando /preco

```mermaid  
sequenceDiagram  
 participant User as Usuário (Telegram)  
 participant N8N as N8N WF\_Bot\_Principal  
 participant IA as Agente IA (LangChain)  
 participant Redis as Redis  
 participant Binance as Binance API  
  
 User->>N8N: /preco BTC  
 N8N->>IA: mensagem  
 IA->>Redis: GET px:spot:BTCBRL  
 alt Cache hit  
 Redis-->>IA: preço BRL  
 else Cache miss  
 IA->>Binance: GET /api/v3/ticker/24hr?symbol=BTCBRL  
 Binance-->>IA: preço + variação 24h  
 IA->>Redis: SETEX px:spot:BTCBRL 10s  
 end  
 IA-->>N8N: resposta formatada  
 N8N-->>User: "BTC: 250k BRL (+2.1% 24h)"  
```

# Diagrama 3 – Monitor de Alertas

```mermaid  
flowchart TD  
 Cron[Cron Job 1-2 min] --> WF[WF\_Monitor\_Alertas]  
 WF --> Supabase[Supabase: SELECT alerts ativos]  
 WF --> Binance[Binance API: preços atuais]  
 WF --> Check{Condição atendida?}  
 Check -->|Sim| Telegram[Enviar alerta]  
 Check -->|Sim| SupabaseUpdate[Update alerta: active=false]  
 Check -->|Não| End[Espera próximo ciclo]  
```

# Diagrama 4 – Portfólio & PnL

```mermaid  
sequenceDiagram  
 participant User as Usuário (Telegram)  
 participant N8N as N8N WF\_Bot\_Principal  
 participant Supabase as Supabase DB  
 participant Binance as Binance API  
  
 User->>N8N: /posicao  
 N8N->>Supabase: SELECT trades do usuário  
 N8N->>Binance: GET preços atuais  
 N8N->>Supabase: Chama função fn\_positions\_cma  
 Supabase-->>N8N: posições + custo médio  
 N8N-->>User: carteira + PnL  
```

# Diagrama 5 – Modelagem de Dados

```mermaid  
classDiagram  
 class User {  
 uuid id  
 bigint chatId  
 string username  
 string language  
 bool isActive  
 }  
 class Alert {  
 bigint id  
 string symbol  
 enum direction (ACIMA, ABAIXO)  
 decimal targetBrl  
 bool active  
 datetime triggeredAt  
 }  
 class Trade {  
 bigint id  
 string symbol  
 enum side (BUY, SELL)  
 decimal qty  
 decimal priceBrl  
 decimal feeBrl  
 datetime ts  
 }  
 class BotLog {  
 bigint id  
 string workflow  
 string action  
 enum level (INFO, WARN, ERROR)  
 string message  
 json details  
 datetime createdAt  
 }  
 class ApiUsage {  
 bigint id  
 enum service  
 string endpoint  
 int count  
 datetime periodStart  
 datetime periodEnd  
 json meta  
 }  
 class ErrorState {  
 bigint id  
 string workflow  
 string node  
 string message  
 json payload  
 datetime createdAt  
 }  
  
 User --> Alert  
 User --> Trade  
 User --> BotLog  
```