FLOWCHART TEXTUAL COMPLETO  POUPE AGORA v2

> Este é o **documento-base do funcionamento da arquitetura inteira**, de ponta a ponta, com todos os caminhos possíveis.

---

# 🟦 (A) ENTRADA — CAMADA DE MENSAGERIA (WhatsApp)

**Usuário envia uma mensagem pelo WhatsApp (texto, áudio, imagem, PDF).**

1. Evolution API recebe a mensagem
2. Evolution chama **Webhook** → **`/webhooks/whatsapp`**
3. Backend recebe o evento e:
    - Normaliza payload
    - Identifica `whatsapp_number`
    - Loga evento em `whatsapp_events_log`
    - Enfileira job → `incoming_message`

---

# 🟦 (B) WORKER: PROCESSAMENTO RAW (Pipeline de Mídia)

O worker lê o job `incoming_message`:

1. Verifica tipo de mídia
2. Executa pipelines:

### Se TEXTO:

- Normaliza
- Continua para Intent Engine

### Se ÁUDIO:

- Faz download do áudio
- Chama OpenAI → transcrição
- Salva transcrição
- Continua para Intent Engine

### Se IMAGEM:

- Baixa a imagem
- OpenAI Vision → extrai texto/valores
- Converte em texto normalizado
- Continua para Intent Engine

### Se DOCUMENTO (PDF/extrato):

- Faz OCR + Vision
- Extrai itens importantes (valor, data, categorias)
- Gera texto normalizado
- Continua para Intent Engine

---

# 🟦 (C) INTENT ENGINE — CLASSIFICADOR DE INTENÇÃO

Entrada: texto normalizado + contexto (intent anterior).

Processo:

1. Consulta Redis:
    - existe intenção pendente?
2. Se existir → usa intenção existente
3. Se não existir:
    - chama “Agente de Intenção” (modelinho leve, GPT-4o-mini)
    - classifica como:
        - `transacaoFinanceira`
        - `consulta`
        - `lembrete`
        - `indefinido`
4. Salva no Redis a intenção atual (TTL = 180s)

**Saída:**

```
{ intent: "...", text: "...", profile_id, family_id }
```

---

# 🟦 (D) ORCHESTRATOR — DIRECIONAMENTO PARA AGENTES

Com base em `intent`:

- `transacaoFinanceira` → Agente de Transações
- `consulta` → Agente de Consulta
- `lembrete` → Agente de Lembretes
- `indefinido` → Mensagem padrão (“não entendi, envie novamente”)

O orquestrador monta:

- contexto
- histórico
- dados do usuário (nome, plano, limites)
- dados financeiros (saldo, metas, categorias, contas etc.)

Chama o agente correto.

---

# 🟦 (E) EXECUÇÃO DO AGENTE (via OpenAI)

Cada agente tem:

- `prompt_de_sistema`
- `tools` autorizadas
- `parameters` (model, temp, max_tokens)
- `exemplos`
- `memória curta` (via Redis)

Fluxo:

1. Orquestrador envia:

    - system prompt (versão ativa)
    - exemplos
    - contexto histórico
    - mensagem do usuário
2. OpenAI retorna:
    - resposta natural
    - chamadas de tools
3. Backend executa tools:
    - CRUD de transações
    - CRUD de lembretes
    - Análises de categorias
    - Cálculos
    - Atualizações de metas
4. Agente monta resposta final amigável

**Saída:**
`{replyText, actionsExecuted, nextIntent}`

---

# 🟦 (F) WORKER DE AÇÕES — PERSISTÊNCIA

As ações executadas são aplicadas:

- Salva transações
- Atualiza categorias
- Cria lembretes

- Atualiza metas
- Atualiza dívidas
- Atualiza score financeiro
- Atualiza logs do agente

Grava em:

- `transactions`
- `categories`
- `reminders`
- `goals`
- `debts`
- `financial_score`
- `ai_logs`

---

# 🟦 (G) RESPOSTA AO USUÁRIO

Worker → fila `send_message`:

1. Monta payload final
2. Chama Evolution API: `sendText / sendMedia`
3. Loga no `whatsapp_events_log`
4. Se intenção finalizada:

- limpa intenção no Redis

---

# 🟦 (H) FLUXOS ADICIONAIS

## 1. Webhook de Pagamento (PerfectPay / Asaas)

1. Evento → `/webhooks/billing`
2. Normaliza
3. Salva `billing_events`
4. Atualiza:
   - `subscriptions`
   - `profiles`
   - `families`
5. Manda WhatsApp de onboard

---

## 2. IA Proativa (Jobs Agendados)

Worker que roda:

- diário
- semanal
- mensal

Gera:

- resumo semanal

- alertas de gasto excessivo
- progresso de metas
- alertas de dívidas
- score POP atualizado

---

### 3. Rotinas Personalizadas (Automations Engine)

Evento → condição → ação:

- nova transação → gasto > limite → enviar alerta
- semana fechada → enviar resumo
- data específica → criar lembrete

---

---

# ✅ 2. FLOWCHART EM MERMAID (PRONTO PARA DIAGRAMA)

Cole isso no Mermaid Live Editor e exporte como PNG/PDF.

```
flowchart TD

%% Entrada WhatsApp
A[Usuário envia mensagem no WhatsApp] --> B(Evolution API)
B --> C[/Webhook /webhooks/whatsapp/]

%% Normalização + Fila
C --> D[Normalizar payload]
D --> E[Log whatsapp_events_log]
E --> F[Enfileirar job incoming_message]
```

```
%% Worker de Mídia
F --> G{Tipo de Mídia?}
G -->|Texto| H[Normalizar texto]
G -->|Áudio| I[Transcrever com OpenAI]
G -->|Imagem| J[Vision OCR + extração]
G -->|PDF| K[OCR + Vision]
H --> L
I --> L
J --> L
K --> L

%% Intent Engine
L --> M{Intenção existente no Redis?}
M -->|Sim| N[Usar intenção atual]
M -->|Não| O[Agente de intenção GPT-4o-mini]
O --> P[Salvar intenção no Redis]
N --> Q
P --> Q

%% Orquestrador
Q --> R{Intent?}
R -->|Transacao| S[Agente de Transações]
R -->|Consulta| T[Agente de Consulta]
R -->|Lembrete| U[Agente de Lembretes]
R -->|Indefinido| V[Responder mensagem padrão]

%% Execução do Agente
S --> W[OpenAI + Tools]
T --> W
U --> W

W --> X[Executar ações: transações, lembretes, metas, categorias]
X --> Y[Persistência: Supabase]
Y --> Z[Log de IA: ai_logs]

%% Resposta
Z --> AA[Fila: send_message]
AA --> AB[Evolution API enviar mensagem]
```

```
AB --> AC[Log outbound no whatsapp_events_log]

%% Limpar intenção
AC --> AD{Finalizar intenção?}
AD -->|Sim| AE[Limpa chave no Redis]
AD -->|Não| AF[Fim]
AE --> AF[Fim]
```

---

# ✅ 3. ARQUITETURA EM JSON (FLOWCHART ESTRUTURADO)

Perfeito para OneDraft / Gemini / Cloud 4.5 aprender a arquitetura toda.

```
{
  "flowchart": {
    "whatsapp_entry": {
      "provider": "Evolution API",
      "webhook": "/webhooks/whatsapp",
      "steps": [
        "Normaliza payload",
        "Loga evento",
        "Enfileira job incoming_message"
      ]
    },
    "media_pipeline": {
      "audio": "Transcrição OpenAI",
      "image": "OpenAI Vision OCR",
      "pdf": "OCR + Vision",
      "text": "Normalização simples"
    },
    "intent_engine": {
      "redis_key": "intent:{whatsapp_number}",
      "if_no_intent": "Classificador GPT-4o-mini",
      "ttl_seconds": 180
    },
```

```
"orchestrator": {
  "routes": {
    "transacaoFinanceira": "Agente de Transações",
    "consulta": "Agente de Consulta",
    "lembrete": "Agente de Lembretes",
    "indefinido": "Mensagem padrão"
  },
  "context": [
    "Dados do usuário",
    "Dados financeiros",
    "Histórico curto",
    "Plano e limites",
    "Config do agente"
  ]
},
"agents_execution": {
  "model": "gpt-4o / gpt-4o-mini",
  "input": [
    "system_prompt",
    "examples",
    "user_message",
    "history"
  ],
  "tools": [
    "CRUD transações",
    "CRUD lembretes",
    "Categorias",
    "Metas",
    "Cálculos"
  ],
  "output": [
    "replyText",
    "actionsExecuted",
    "nextIntent"
  ]
},
"actions_processing": {
  "tables": [
```

```
      "transactions",
      "categories",
      "goals",
      "debts",
      "reminders",
      "financial_score"
    ],
    "logs": "ai_logs"
  },
  "whatsapp_reply": {
    "queue": "send_message",
    "provider": "Evolution API",
    "cleanup": "Limpa intenção se finalizada"
  },
  "billing_webhook": {
    "providers": ["PerfectPay", "Asaas"],
    "steps": [
      "Normalizar evento",
      "Salvar billing_events",
      "Atualizar subscriptions",
      "Atualizar profiles",
      "Enviar boas-vindas via WhatsApp"
    ]
  },
  "proactive_ia": {
    "schedule": ["daily", "weekly"],
    "actions": [
      "Resumo semanal",
      "Alertas de gastos",
      "Progresso de metas",
      "Status de dívidas",
      "Atualização do Score POP"
    ]
  },
  "automations": {
    "trigger_condition_action": true,
    "storage": "user_automations",
    "examples": [
```

```
        "Gasto > limite → alerta",
        "Fechamento semanal → resumo",
        "Data específica → lembrete"
      ]
    }
  }
}
```