📐 Arquitetura da API - Bot de Cobrança Multi-Empresa

🎯 Visão Geral

Sistema de conversação inteligente baseado em LangGraph que processa mensagens de clientes, analisa sentimento e intenção, busca conhecimento contextual (RAG) e decide quando notificar corretores humanos.

🔄 Fluxo Completo de Processamento

```
Cliente → API → LangGraph (7 nós) → Resposta Estruturada
                  ↓
              MongoDB (histórico + RAG)
                  ↓
              OpenAI (embeddings + LLM)
```

📡 Endpoint Principal: POST /chat

URL

```
POST http://localhost:8000/chat
```

Headers

```
Content-Type: application/json
```

📥 REQUEST - Estrutura de Entrada

Schema Completo

json

```
{
  "company_id": "string",
  "company_name": "string",
  "customer_id": "string",
  "message": "string",
  "start_chat": "string | null",
  "customer_context": {
    "name": "string",
    "policies": [
      {
        "policy_number": "string",
        "total_due_value": "float",
        "total_due_installments": "integer",
        "due_installments": [
          {
            "number": "integer",
            "value": "float",
            "due_date": "YYYY-MM-DD",
            "days_overdue": "integer"
          }
        ]
      }
    ],
    "total_due_value": "float",
    "total_due_installments": "integer",
    "consultant_name": "string"
  }
}
```

Descrição dos Campos

Campo	Tipo	Obrigatório	Descrição
`company_id`	string	✅	Identificador único da empresa
`company_name`	string	✅	Nome da empresa para personalizar o prompt
`customer_id`	string	✅	Identificador único do cliente (usado como session_id)
`message`	string	✅	Mensagem atual do cliente (mín: 1 char)
`start_chat`	string | null	❌	Mensagem inicial enviada por sistema externo (apenas 1ª interação)
`customer_context`	object	✅	Contexto completo do cliente
`customer_context.name`	string	✅	Nome do cliente
`customer_context.policies`	array	✅	Lista de apólices do cliente
`customer_context.policies[].policy_number`	string	✅	Número da apólice
`customer_context.policies[].total_due_value`	float	✅	Valor total devido da apólice
`customer_context.policies[].total_due_installments`	integer	✅	Total de parcelas em atraso
			Detalhes das

customer_context.policies[].due_installments	array	✅	parcelas em atraso
customer_context.policies[].due_installments[].number	integer	✅	Número da parcela
customer_context.policies[].due_installments[].value	float	✅	Valor da parcela
customer_context.policies[].due_installments[].due_date	date	✅	Data de vencimento (YYYY-MM-DD)
customer_context.policies[].due_installments[].days_overdue	integer	✅	Dias em atraso
customer_context.total_due_value	float	✅	Valor total devido (todas as apólices)
customer_context.total_due_installments	integer	✅	Total de parcelas em atraso (todas as apólices)
customer_context.consultant_name	string	✅	Nome do consultor/ corretor responsável

Exemplo de Request - Primeira Interação

json


```
{
  "company_id": "metlife_001",
  "company_name": "Metlife Seguros",
  "customer_id": "cliente_12345",
  "message": "Oi, sobre o que é essa cobrança?",
  "start_chat": "Olá João! Identificamos uma pendência na sua apólice 397910 no valor de R$ 1.500,50. Pode
  "customer_context": {
    "name": "João Silva",
    "policies": [
      {
        "policy_number": "397910",
        "total_due_value": 1500.50,
        "total_due_installments": 3,
        "due_installments": [
          {
            "number": 1,
            "value": 500.17,
            "due_date": "2024-08-30",
            "days_overdue": 90
          },
          {
            "number": 2,
            "value": 500.17,
            "due_date": "2024-09-30",
            "days_overdue": 60
          },
          {
            "number": 3,
            "value": 500.16,
            "due_date": "2024-10-30",
            "days_overdue": 30
          }
        ]
      }
    ],
    "total_due_value": 1500.50,
    "total_due_installments": 3,
    "consultant_name": "Maria Consultora"
  }
}
```


Exemplo de Request - Interação Subsequente

json


```
{
  "company_id": "metlife_001",
  "company_name": "Metlife Seguros",
  "customer_id": "cliente_12345",
  "message": "Vou pagar! Me envia o boleto",
  "start_chat": null,
  "customer_context": {
    "name": "João Silva",
    "policies": [...],
    "total_due_value": 1500.50,
    "total_due_installments": 3,
    "consultant_name": "Maria Consultora"
  }
}
```

📤 RESPONSE - Estrutura de Saída

Schema Completo

json


```
{
  "session_id": "string",
  "response": {
    "reply": "string",
    "notify": "boolean",
    "intent": "PAYMENT | REFUSED | CANCEL | OTHER | NEUTRAL",
    "summary": "string | null",
    "handoff": "string | null",
    "status": "string",
    "sentiment": "string",
    "update_kanban_status": "string | null"
  },
  "metadata": {
    "tools_used": ["string"],
    "rag_items_retrieved": "integer",
    "sentiment": "string",
    "intent": "string",
    "tokens_used": {
      "prompt": "integer",
      "completion": "integer",
      "total": "integer"
    },
    "error": "string | null"
  }
}
```


Descrição dos Campos

`response` (objeto principal)

Campo	Tipo	Valores	Descrição
`reply`	string	-	Mensagem para o cliente (10-500 chars)
`notify`	boolean	true/false	Se deve notificar o corretor
`intent`	enum	PAYMENT, REFUSED, CANCEL, OTHER, NEUTRAL	Intenção classificada do cliente
`summary`	string | null	-	Resumo da interação (máx: 200 chars)
`handoff`	string | null	-	Instrução para o corretor (máx: 250 chars)
`status`	enum	Ver tabela abaixo	Status da conversa
`sentiment`	enum	Ver tabela abaixo	Sentimento detectado
`update_kanban_status`	enum | null	Ver tabela abaixo	Novo status do Kanban

`metadata` (informações técnicas)

Campo	Tipo	Descrição
`tools_used`	array[string]	Lista de tools executadas (sentiment, intent)
`rag_items_retrieved`	integer	Número de FAQs recuperadas do RAG
`sentiment`	string	Sentimento detectado (cópia para conveniência)
`intent`	string	Intenção detectada (cópia para conveniência)
`tokens_used`	object	Contadores de tokens OpenAI
`tokens_used.prompt`	integer	Tokens do prompt
`tokens_used.completion`	integer	Tokens da resposta
`tokens_used.total`	integer	Total de tokens
`error`	string | null	Mensagem de erro (se houver)

📊 Enumerações (Enums)

`intent` - Intenção do Cliente

Valor	Descrição	Exemplo
`PAYMENT`	Cliente quer pagar, aceita boleto	"Vou pagar amanhã", "Pode enviar o boleto"
`REFUSED`	Cliente recusa pagamento, contesta	"Não vou pagar", "Já quitei isso"
`CANCEL`	Cliente quer cancelar apólice/seguro	"Quero cancelar minha apólice"
`OTHER`	Outros assuntos (mudança, dados, sinistro)	"Preciso alterar meu endereço"
`NEUTRAL`	Perguntas, dúvidas, indeciso	"Como faço para pagar?", "Quanto está devendo?"

`status` - Status da Conversa

Valor	Descrição
`OUTBOUND_SENT`	Primeira mensagem enviada (outbound)
`CONVERSING`	Bot está conversando com o cliente
`CUSTOMER_READY`	Cliente pronto para pagar
`NOTIFIED_CONSULTANT`	Corretor foi notificado
`CLOSED_SUCCESS`	Conversa fechada com sucesso
`CLOSED_NO_RESPONSE`	Conversa fechada sem resposta
`CLOSED_REFUSED`	Conversa fechada com recusa

`sentiment` - Sentimento do Cliente

Valor	Descrição
`positivo`	Cliente satisfeito, cooperativo
`neutro`	Cliente neutro, apenas informativo
`negativo`	Cliente insatisfeito mas controlado
`raiva`	Cliente irritado, agressivo
`ansioso`	Cliente preocupado, urgente
`confuso`	Cliente perdido, não entende
`envergonhado`	Cliente constrangido pela dívida
`triste`	Cliente em dificuldade emocional

`update_kanban_status` - Status do Kanban

Valor	Quando Usar
`Identificado`	Cliente identificado no sistema
`Contatado`	Primeira interação realizada
`Negociação`	Cliente em negociação, fazendo perguntas
`Promessa`	Cliente prometeu pagar em data específica
`Acordo`	Cliente aceitou condições de pagamento
`Handoff Consultor`	Cliente pediu atendimento humano
`Quitado`	Débito quitado
`Quebra de promessa/acordo`	Cliente não cumpriu promessa
`Recusa`	Cliente recusou pagamento
`Outros Assuntos`	Cliente quer tratar de cancelamento ou outros assuntos

🎯 Cenários de Resposta

Cenário 1: Cliente Quer Pagar (PAYMENT)

Request:

```json
{
  "message": "Vou pagar! Pode enviar o boleto",
  ...
}
```

Response:

json


```
{
  "session_id": "cliente_12345",
  "response": {
    "reply": "Perfeito João! Vou notificar o consultor para enviar o boleto imediatamente.",
    "notify": true,
    "intent": "PAYMENT",
    "summary": "Cliente comprometeu-se a pagar",
    "handoff": "Enviar boleto/pix para João Silva no valor de R$ 1.500,50 referente à apólice 397910",
    "status": "CUSTOMER_READY",
    "sentiment": "positivo",
    "update_kanban_status": "Acordo"
  },
  "metadata": {
    "tools_used": ["sentiment", "intent"],
    "rag_items_retrieved": 2,
    "sentiment": "positivo",
    "intent": "PAYMENT",
    "tokens_used": {
      "prompt": 2800,
      "completion": 180,
      "total": 2980
    }
  }
}
```


Cenário 2: Cliente Recusa (REFUSED)

Request:

json


```
{
  "message": "Não vou pagar isso! Já cancelei esse seguro",
  ...
}
```


Response:

```json
{
  "session_id": "cliente_12345",
  "response": {
    "reply": "Entendo sua frustração. Vou encaminhar sua situação para o consultor analisar o histórico.",
    "notify": true,
    "intent": "REFUSED",
    "summary": "Cliente recusa pagamento alegando cancelamento prévio",
    "handoff": "Analisar recusa do cliente João Silva que alega ter cancelado. Verificar histórico",
    "status": "NOTIFIED_CONSULTANT",
    "sentiment": "raiva",
    "update_kanban_status": "Recusa"
  },
  "metadata": {
    "tools_used": ["sentiment", "intent"],
    "rag_items_retrieved": 1,
    "sentiment": "raiva",
    "intent": "REFUSED",
    "tokens_used": {
      "prompt": 2750,
      "completion": 160,
      "total": 2910
    }
  }
}
```

Cenário 3: Cliente Quer Cancelar (CANCEL)

Request:

```json
{
  "message": "Quero cancelar minha apólice",
  ...
}
```

Response:

```json
{
  "session_id": "cliente_12345",
  "response": {
    "reply": "Entendo. Vou encaminhar sua solicitação de cancelamento para análise do consultor.",
    "notify": true,
    "intent": "CANCEL",
    "summary": "Cliente deseja cancelar apólice",
    "handoff": "Cliente João Silva deseja cancelar apólice 397910. Direcionar para setor de retenção",
    "status": "NOTIFIED_CONSULTANT",
    "sentiment": "neutro",
    "update_kanban_status": "Outros Assuntos"
  },
  "metadata": {
    "tools_used": ["sentiment", "intent"],
    "rag_items_retrieved": 0,
    "sentiment": "neutro",
    "intent": "CANCEL",
    "tokens_used": {
      "prompt": 2650,
      "completion": 145,
      "total": 2795
    }
  }
}
```

Cenário 4: Cliente com Dúvida (NEUTRAL)

Request:

```json
{
  "message": "Como faço para gerar um boleto?",
  ...
}
```

Response:

json


```
{
  "session_id": "cliente_12345",
  "response": {
    "reply": "O boleto será enviado pelo seu consultor via email ou WhatsApp. Você também pode solicitar u
    "notify": false,
    "intent": "NEUTRAL",
    "summary": "Cliente perguntou sobre geração de boleto",
    "handoff": null,
    "status": "CONVERSING",
    "sentiment": "confuso",
    "update_kanban_status": "Negociação"
  },
  "metadata": {
    "tools_used": ["sentiment", "intent"],
    "rag_items_retrieved": 5,
    "sentiment": "confuso",
    "intent": "NEUTRAL",
    "tokens_used": {
      "prompt": 3100,
      "completion": 120,
      "total": 3220
    }
  }
}
```


Cenário 5: Outros Assuntos (OTHER)

Request:

json


```
{
  "message": "Preciso atualizar meu endereço",
  ...
}
```


Response:

json


```
{
  "session_id": "cliente_12345",
  "response": {
    "reply": "Entendo que você precisa atualizar seu endereço. Vou encaminhar para o consultor que poderá
    "notify": true,
    "intent": "OTHER",
    "summary": "Cliente deseja atualizar endereço cadastral",
    "handoff": "Cliente João Silva deseja atualizar endereço. Direcionar para setor de cadastro",
    "status": "NOTIFIED_CONSULTANT",
    "sentiment": "neutro",
    "update_kanban_status": "Outros Assuntos"
  },
  "metadata": {
    "tools_used": ["sentiment", "intent"],
    "rag_items_retrieved": 1,
    "sentiment": "neutro",
    "intent": "OTHER",
    "tokens_used": {
      "prompt": 2700,
      "completion": 155,
      "total": 2855
    }
  }
}
```

⚙️ Processamento Interno (LangGraph)

Fluxo de 7 Nós

```
1. LOAD_CONTEXT
   ↓ Carrega histórico + RAG + start_chat

2. SENTIMENT
   ↓ Analisa sentimento (8 categorias)

3. INTENT
   ↓ Analisa intenção (5 categorias)

4. VALIDATE
   ↓ Valida que tools foram executadas

5. RESPOND
   ↓ LLM gera resposta estruturada

6. PROCESS_DECISION
   ↓ Valida regras de negócio + preenche handoff

7. SAVE
   ↓ Persiste no MongoDB

END
```

Regras de Negócio (Process Decision)

Regra 1: notify=true → intent ≠ NEUTRAL

Quando `notify: true`, o campo `intent` **NUNCA** pode ser `NEUTRAL`.

Regra 2: Mapeamento de Situações

Situação	notify	intent	status	kanban
Cliente confirma pagamento	true	PAYMENT	CUSTOMER_READY	Acordo/Promessa
Cliente recusa	true	REFUSED	NOTIFIED_CONSULTANT	Recusa
Cliente quer cancelar	true	CANCEL	NOTIFIED_CONSULTANT	Outros Assuntos
Cliente fala outros assuntos	true	OTHER	NOTIFIED_CONSULTANT	Outros Assuntos
Cliente pede humano	true	OTHER	NOTIFIED_CONSULTANT	Handoff Consultor
Cliente tem dúvida	false	NEUTRAL	CONVERSING	Negociação

Regra 3: Handoff Obrigatório

Quando `notify: true`, o campo `handoff` **SEMPRE** é preenchido com instrução específica.

🔐 Autenticação e Rate Limiting

Autenticação

Atualmente sem autenticação. Recomenda-se implementar:

- API Key no header: `X-API-Key: <key>`
- OAuth 2.0 para integrações enterprise

Rate Limiting

Não implementado. Recomenda-se:

- 100 requests/minuto por `company_id`
- 10 requests/segundo por `customer_id`

❌ Códigos de Erro

HTTP Status Codes

Código	Descrição
`200`	Sucesso
`400`	Bad Request - Validação falhou
`404`	Not Found - Recurso não encontrado
`422`	Unprocessable Entity - Schema inválido
`500`	Internal Server Error - Erro no servidor

Exemplo de Erro 422

```json
{
  "detail": [
    {
      "type": "string_too_short",
      "loc": ["body", "message"],
      "msg": "String should have at least 1 character",
      "input": "",
      "ctx": {
        "min_length": 1
      }
    }
  ]
}
```

Exemplo de Erro 500

```json
{
  "detail": "Erro no processamento: Tools não foram validadas. Fluxo interrompido."
}
```

🔄 Gestão de Sessões

TTL (Time To Live)

- Sessões expiram após 30 dias (configurável via `SESSION_TTL_DAYS`)
- TTL é definido na **criação** da sessão
- MongoDB deleta automaticamente via índice TTL

Estrutura no MongoDB

json


```
{
  "_id": ObjectId("..."),
  "session_id": "cliente_12345",
  "company_id": "metlife_001",
  "messages": [
    {
      "role": "assistant",
      "content": "Olá João! Identificamos...",
      "timestamp": "2024-01-15T10:00:00Z",
      "metadata": {
        "type": "outbound_initial",
        "source": "external_system"
      }
    },
    {
      "role": "user",
      "content": "Oi, sobre o que é?",
      "timestamp": "2024-01-15T10:05:00Z",
      "metadata": {}
    }
  ],
  "rag_context_used": [
    {
      "question": "Como gerar boleto?",
      "relevance_score": 0.89,
      "used_at": "2024-01-15T10:05:00Z"
    }
  ],
  "summary": {
    "total_interactions": 5,
    "sentiment_history": ["confuso", "positivo"],
    "intent_history": ["NEUTRAL", "PAYMENT"],
    "last_kanban_status": "Acordo",
    "rag_hits": 3
  },
  "customer_context": {...},
  "created_at": "2024-01-15T10:00:00Z",
  "updated_at": "2024-01-15T10:30:00Z",
  "expires_at": "2024-02-14T10:00:00Z"
}
```

📊 Métricas e Observabilidade

Logs Estruturados

Cada nó do grafo registra logs:

```
[LOAD_CONTEXT] Iniciando para customer cliente_12345
[LOAD_CONTEXT] Histórico: 10 msgs, Recente: 4 msgs
[LOAD_CONTEXT] RAG: 3 FAQs recuperadas
[SENTIMENT] Resultado: confuso (score: 70, confiança: média)
[INTENT] Resultado: NEUTRAL - Cliente pediu mais informações
[VALIDATE] ✅ Tools validadas com sucesso
[AGENT_RESPOND] Tokens usados: 2500 + 150 = 2650
[PROCESS_DECISION] ✅ Decisão final: notify=False, intent=NEUTRAL
[SAVE_SESSION] ✅ Sessão salva com sucesso
```

Métricas Recomendadas

- Latência média por request
- Taxa de notificações (`notify=true`)
- Distribuição de intents (PAYMENT, REFUSED, etc)
- Distribuição de sentimentos
- Taxa de uso do RAG
- Tokens consumidos (custos OpenAI)

💡 Boas Práticas de Integração

1. Idempotência

Use `customer_id` único para evitar duplicação de sessões.

2. Retry Logic

Implemente retry exponencial para erros 500:

```
1s → 2s → 4s → 8s (máx 3 tentativas)
```

3. Timeout

Configure timeout de **30 segundos** para requests.

4. Validação de Schema

Valide o schema antes de enviar para evitar 422.

5. Tratamento de `notify=true`

Quando `notify=true`, seu sistema deve:

1. Enviar webhook/notificação para o corretor
2. Usar o campo `handoff` como instrução
3. Atualizar o Kanban com `update_kanban_status`
4. Registrar o `intent` para analytics

6. Uso do `start_chat`

- **Primeira interação:** Preencha com mensagem enviada pelo sistema externo
- **Interações subsequentes:** Sempre `null`
- **Validação:** Sistema ignora `start_chat` se histórico não estiver vazio

---

🔗 Endpoints Relacionados

Knowledge Base (RAG)

- `POST /knowledge` - Criar FAQ
- `GET /knowledge` - Listar FAQs
- `PUT /knowledge/{id}` - Atualizar FAQ
- `DELETE /knowledge/{id}` - Deletar FAQ
- `POST /knowledge/bulk` - Criação em massa

Sessões

- `GET /sessions/{customer_id}` - Obter histórico completo
- `DELETE /sessions/{customer_id}` - Reset de sessão

Saúde

- `GET /health` - Status da API

📞 Suporte

Para dúvidas sobre a arquitetura ou integração:

- Documentação interativa: `http://localhost:8000/docs`
- Abra uma issue no repositório

Versão: 1.0.0
Última Atualização: Janeiro 2025