

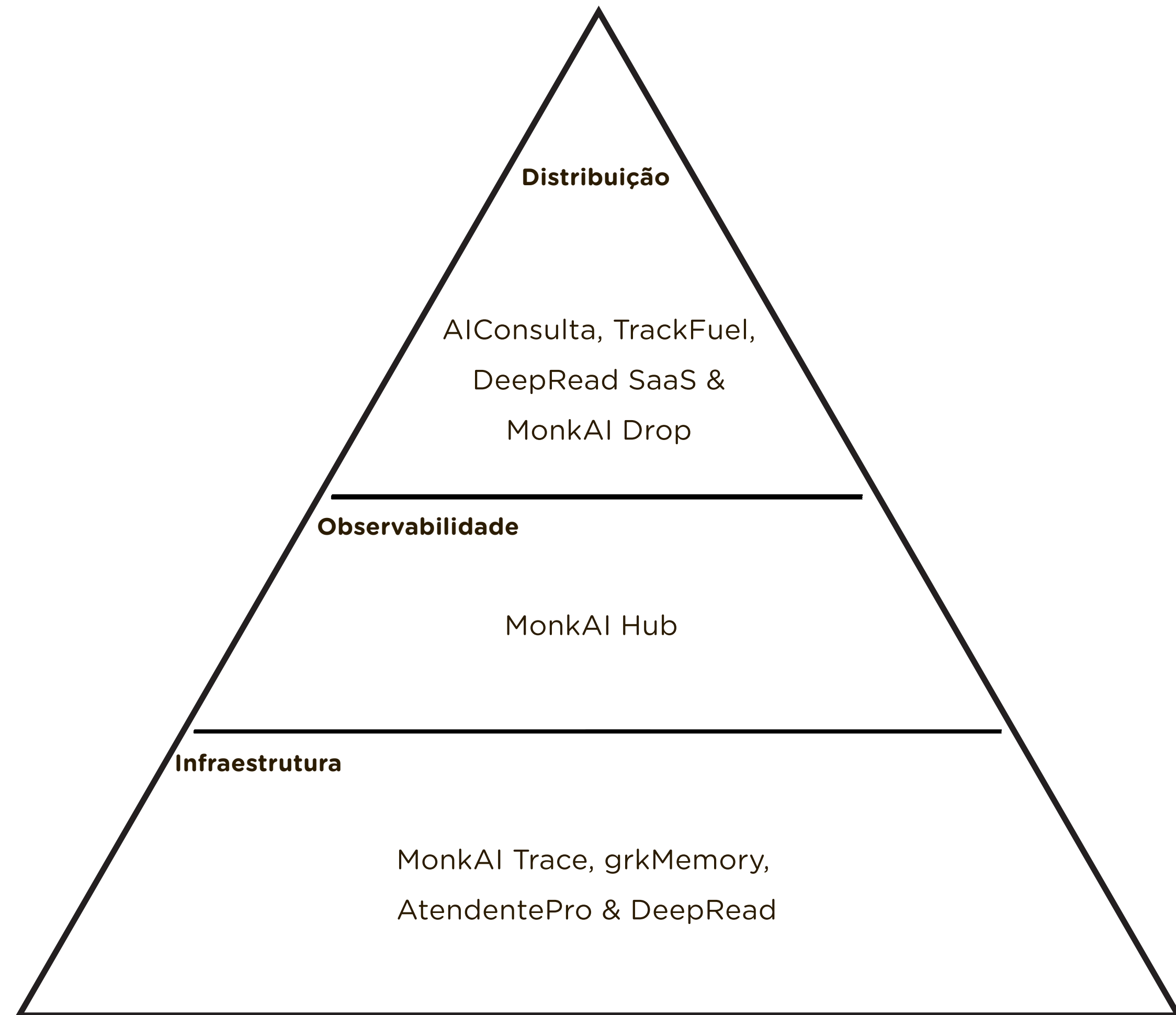


MONKAI

O que é MonkAI?

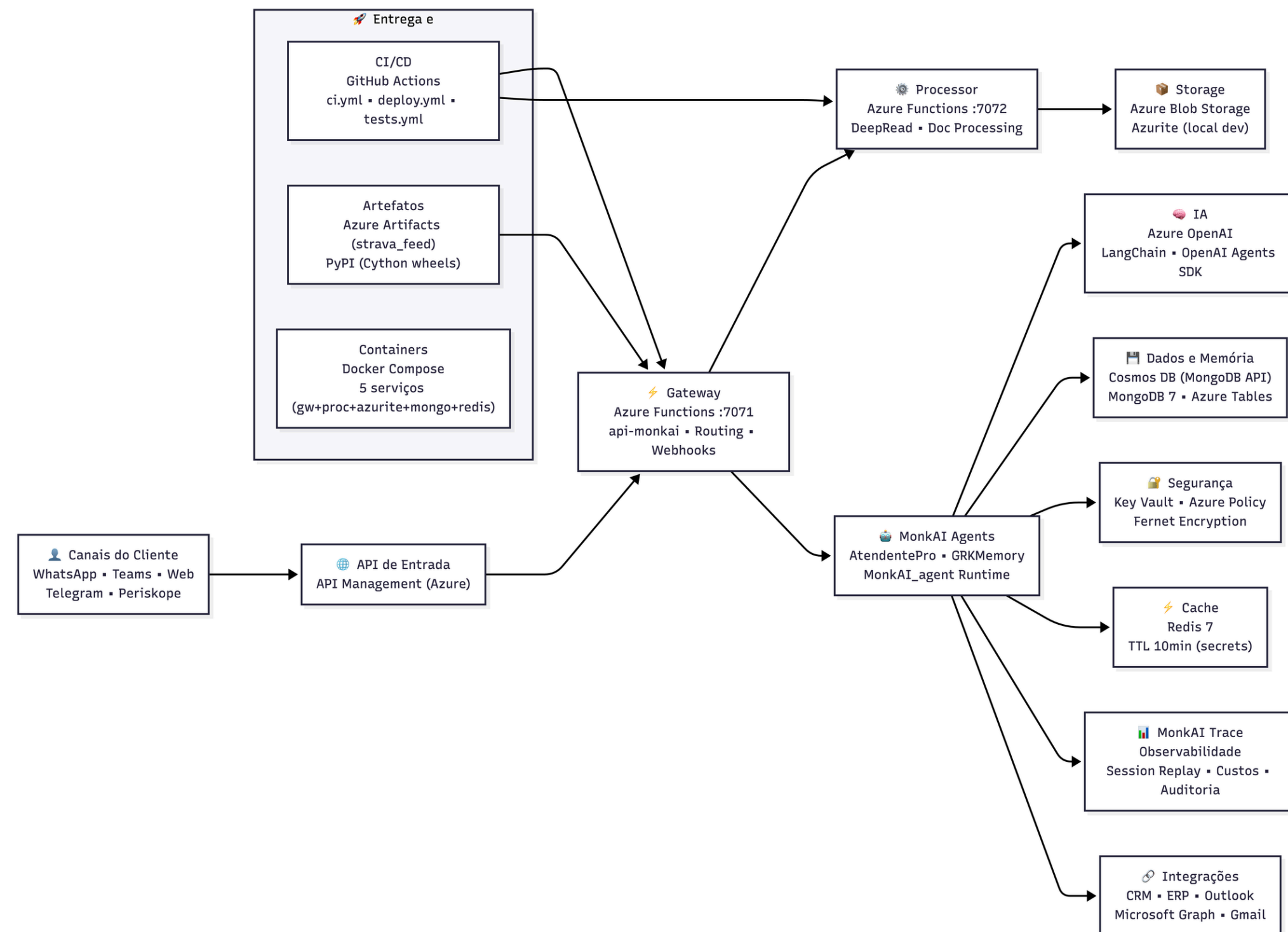
MonkAI é uma deeptech que oferece uma plataforma Agent-as-a-Service (AaaS) completa para empresas.

Com infraestrutura, produtos prontos ou customizáveis e governança, para operar e escalar agentes de IA em produção.



MonkAi Arquitetura (Multi Tenant)

A MonkAI roda no Azure, conectando canais (WhatsApp, Teams, Web) aos sistemas do cliente. Usa Azure OpenAI para automações, Cosmos e Blob para dados e Key Vault para segurança, com observabilidade ponta a ponta no MonkAI Trace (qualidade e custos).



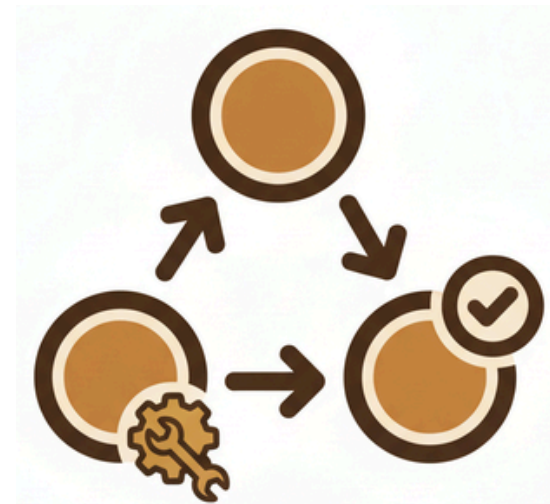
OBSERVABILIDADE

MonkAI Hub

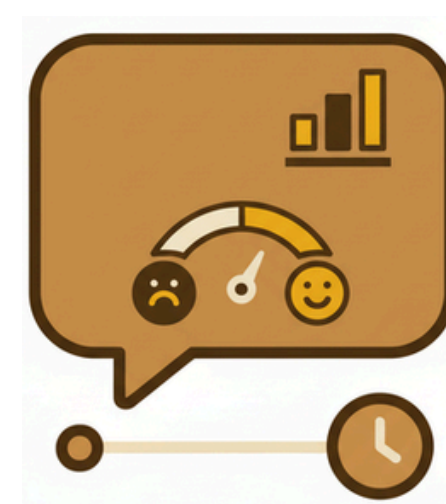
O Hub oferece duas formas de visualização que garantem autonomia e confiança tanto para desenvolvedores quanto para a área de negócio.



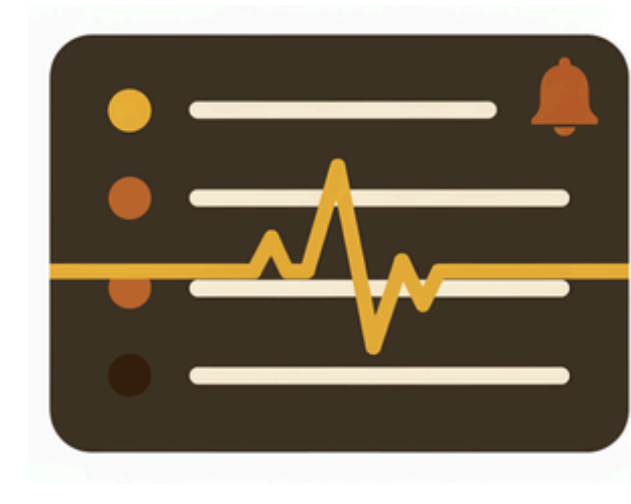
**Teste
validação**



**Fluxo
agentes**



**Conversas
insights**



**Monitoramento
observabilidade**

MonkAI Hub

Transforma agentes em sistemas auditáveis, otimizáveis e confiáveis — com controle total sobre os dados e as decisões.

The screenshot displays the MonkAI Hub interface, divided into a left sidebar and a main content area.

Left Sidebar:

- MONKAI logo
- MODO DE VISUALIZAÇÃO: Negócio (selected) and Dev <>
- Conversas & Custos (button)
- Principal: Monitoramento (selected), Tickets, Perfil, Configurações
- Ajuda: Suporte
- Administração: Painel Admin
- Créditos: -43.999
- Sair (button)

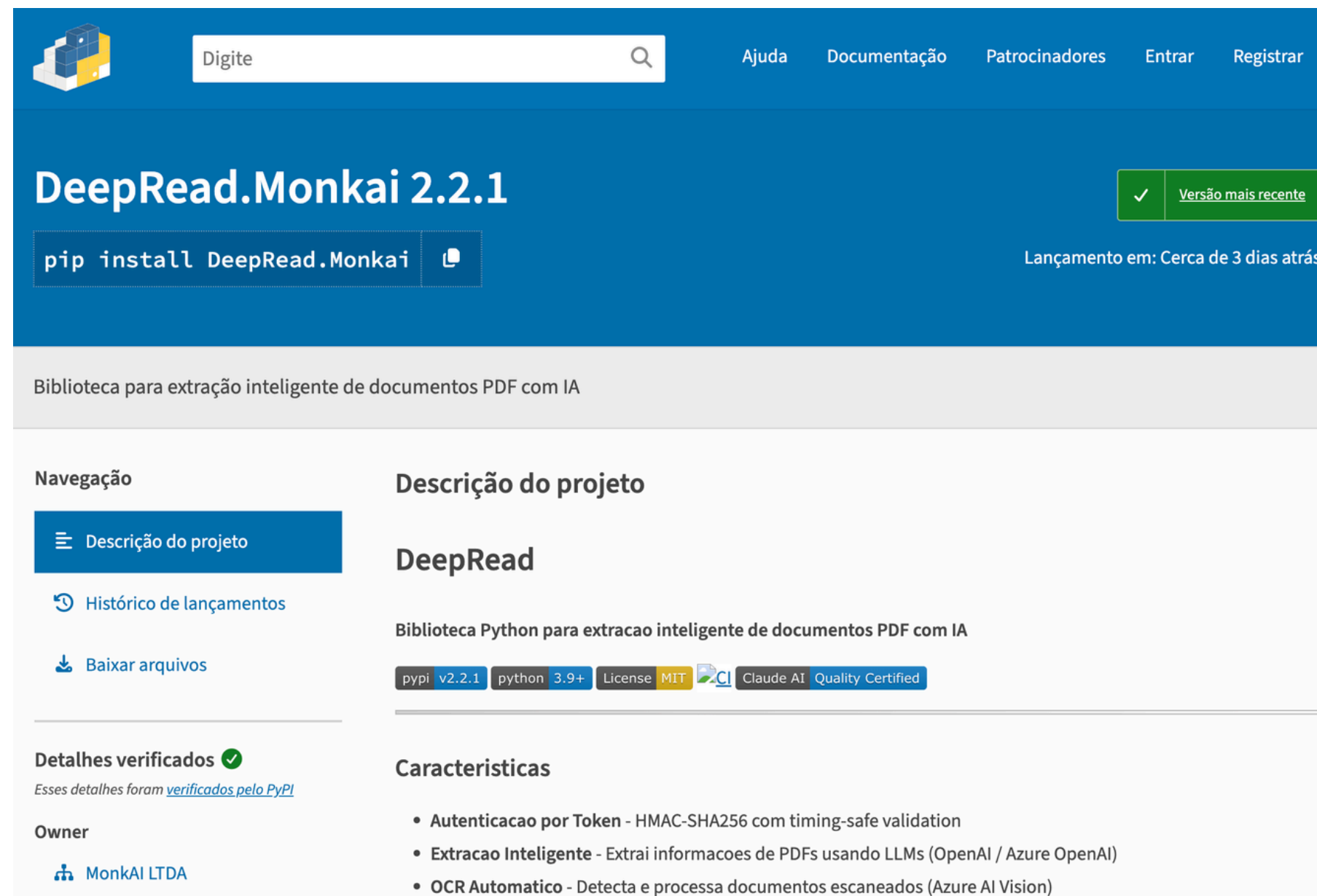
Main Content Area:

- Monitoramento MonkAI** (Section Header)
- Sub-header: Acompanhe métricas de negócio e performance dos seus agentes
- Alerts: Atualizado há 30s, Configurar Alertas
- Filtros** (Section Header): Configure os filtros para personalizar a visualização de dados
- Search: Buscar por conteúdo ou session ID... (7 dias, Todos agentes...)
- Namespace: Todos os namespaces
- Mais Filtros (button)
- Navigation: Conversas (selected), Volumetria, Analytics de Negócio
- Histórico de Conversas** (Section Header): Todas as conversas registradas entre usuários e agentes de IA (Agrupamento: 120s)
- Options: Simplificada, Avançada, Exportar CSV
- Conversation 1:** Sessão: vivo-sup... (Felipe De Oliveira Sposito (GV), web) 03/03/2026, 17:47 - 03/03/2026, 17:47 (Ver Detalhes)
 - User: Felipe De Oliveira Sposito (GV) Me dê um resumo executivo da minha carteira
 - Agent Status: Olá! Verifiquei o status da sua carteira e identifiquei um detalhe importante que impede a visualização dos dados no momento: ⚠️ **Aviso do Sistema** **Mensagem** | Perfil sem código GN configurado **Status** | Acesso Restrito **Ação Necessária** | Contatar Administrador 📧...
- Conversation 2:** Sessão: vivo-sup... (Teste consulta flexivel, web) 03/03/2026, 17:11 - 03/03/2026, 17:12 (Ver Detalhes)

INFRAESTRUTURA

DeepRead

Biblioteca Python para extração inteligente de documentos PDF com IA e validação de poderes e informações no contrato junto à Receita Federal.

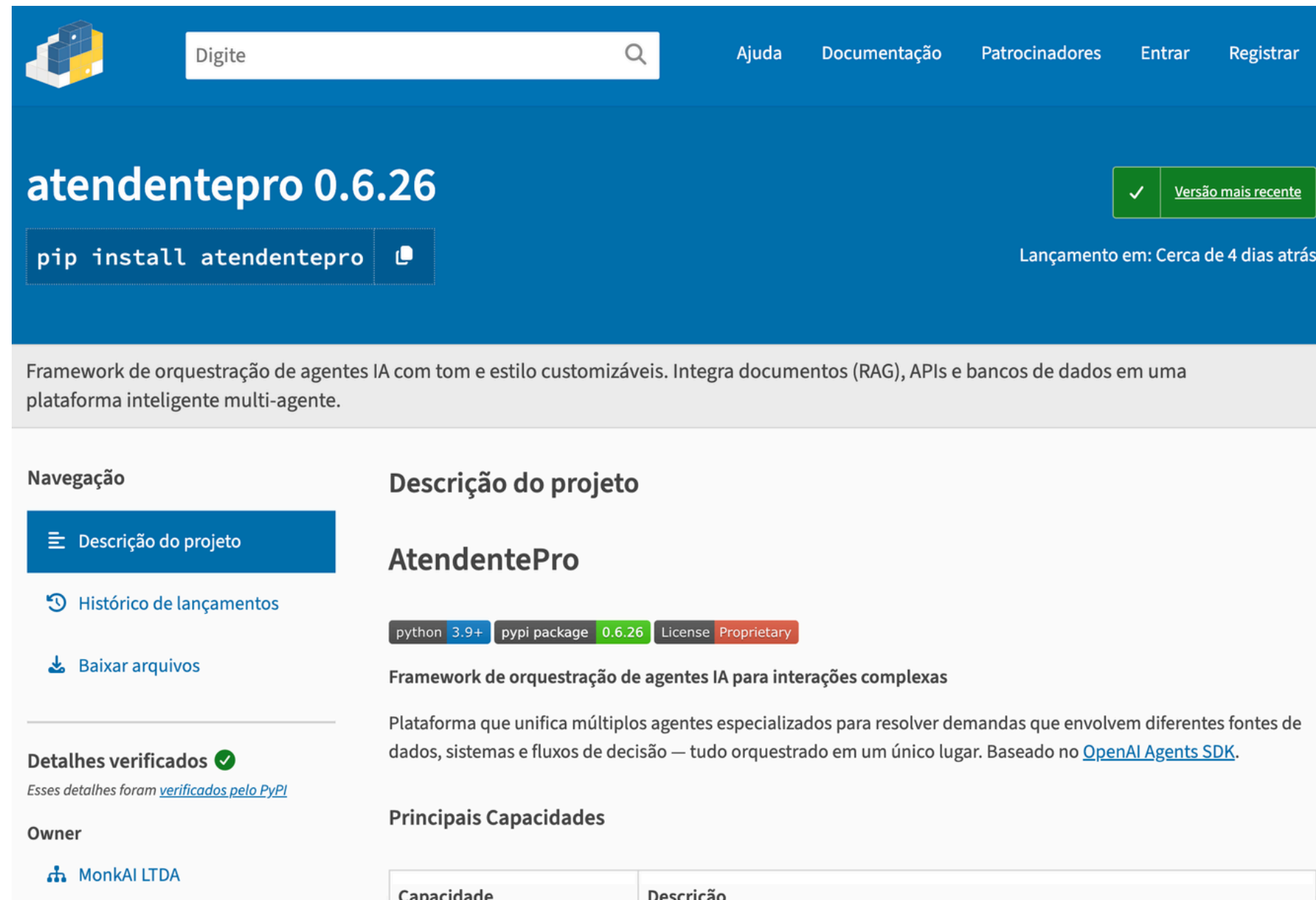


The screenshot shows the PyPI project page for DeepRead.Monkai 2.2.1. The page has a blue header with a search bar and navigation links: Ajuda, Documentação, Patrocinadores, Entrar, and Registrar. The main content area is white with a blue sidebar on the left. The sidebar contains a 'Navegação' section with links for 'Descrição do projeto', 'Histórico de lançamentos', and 'Baixar arquivos'. Below this is a 'Detalhes verificados' section with a green checkmark and a note that details were verified by PyPI. The 'Owner' section lists MonkAI LTDA. The main content area features a 'Descrição do projeto' section with the title 'DeepRead' and a subtitle 'Biblioteca Python para extração inteligente de documentos PDF com IA'. Below the subtitle are several badges: 'pypi v2.2.1', 'python 3.9+', 'License MIT', 'Claude AI', and 'Quality Certified'. The 'Características' section lists three features: 'Autenticacao por Token - HMAC-SHA256 com timing-safe validation', 'Extração Inteligente - Extrai informacoes de PDFs usando LLMs (OpenAI / Azure OpenAI)', and 'OCR Automatico - Detecta e processa documentos escaneados (Azure AI Vision)'. A green button labeled 'Versão mais recente' is visible in the top right corner of the main content area, and a note indicates the release was 'Cerca de 3 dias atrás'.

<https://pypi.org/project/DeepRead.Monkai/>

AtendentePro

Framework de orquestração de agentes de IA para interações complexas. Plataforma que unifica múltiplos agentes especializados para resolver demandas que envolvem diferentes fontes de dados, sistemas e fluxos de decisão — tudo orquestrado em um único lugar.

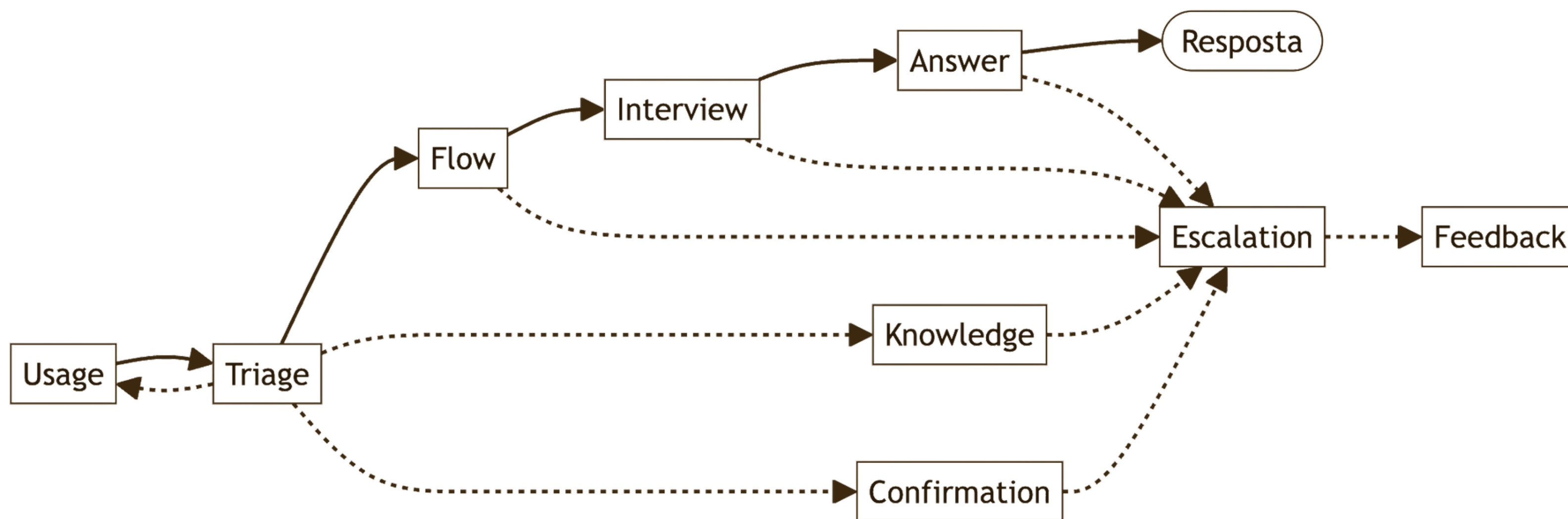


The screenshot shows the PyPI project page for AtendentePro. The header includes a search bar with the text "Digite" and a magnifying glass icon, and navigation links for "Ajuda", "Documentação", "Patrocinadores", "Entrar", and "Registrar". The main section features the project name "atendente pro 0.6.26" in a large font, with a green checkmark and the text "Versão mais recente" to its right. Below this is a code block containing the command "pip install atendente pro" and a copy icon. To the right of the code block, it says "Lançamento em: Cerca de 4 dias atrás". A descriptive paragraph follows: "Framework de orquestração de agentes IA com tom e estilo customizáveis. Integra documentos (RAG), APIs e bancos de dados em uma plataforma inteligente multi-agente." The page is divided into two columns. The left column, titled "Navegação", contains links for "Descrição do projeto" (highlighted), "Histórico de lançamentos", and "Baixar arquivos". Below this is a "Detalhes verificados" section with a green checkmark and the text "Esses detalhes foram verificados pelo PyPI". The "Owner" section lists "MonkAI LTDA" with a GitHub icon. The right column, titled "Descrição do projeto", features the project name "AtendentePro" and a metadata bar showing "python 3.9+", "pypi package 0.6.26", and "License Proprietary". Below this is the project description: "Framework de orquestração de agentes IA para interações complexas" and "Plataforma que unifica múltiplos agentes especializados para resolver demandas que envolvem diferentes fontes de dados, sistemas e fluxos de decisão — tudo orquestrado em um único lugar. Baseado no [OpenAI Agents SDK](#)." The "Principais Capacidades" section is partially visible at the bottom, showing a table with columns "Capacidade" and "Descrição".

<https://pypi.org/project/atendente pro/>

AtendentePro

A operação começa com a triagem da intenção do usuário e, a partir disso, o atendimento pode seguir para fluxos guiados, coleta estruturada de dados, respostas objetivas, consulta à base de conhecimento, confirmações ou escalonamento humano. Esse modelo reduz atrito, melhora a experiência e aumenta a taxa de resolução automática.



Linhas tracejadas: rotas alternativas comuns (roteamento, dúvidas, encerramento, tickets).

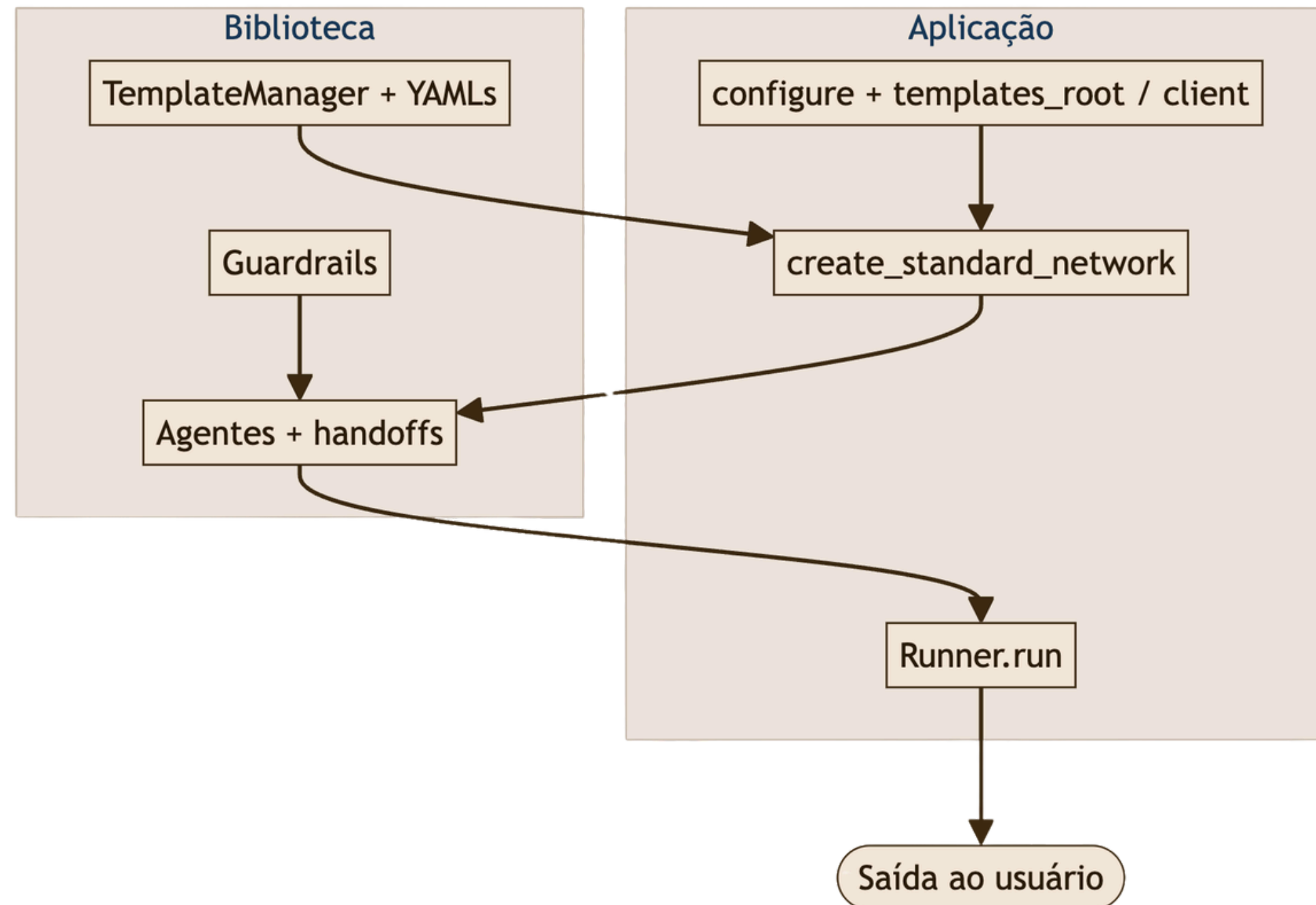
AtendentePro

Cada agente do AtendentePro possui uma função específica dentro da operação, como triagem, condução do fluxo, entrevistas estruturadas, resposta final, consulta de conhecimento, confirmação de dados, onboarding, suporte de uso, escalonamento e feedback. Essa divisão de responsabilidades permite maior qualidade no atendimento e mais controle sobre cada etapa da jornada.

Agente	Papel
Triage	Entrada: entender intenção e delegar.
Flow	Fluxo conversacional guiado.
Interview	Coleta estruturada de dados.
Answer	Respostas finais sintetizadas.
Knowledge	RAG / base de conhecimento.
Confirmation	Validações e confirmações.
Usage	Como usar o produto / tutoriais.
Onboarding	Boas-vindas e cadastro (opcional).
Escalation	Encaminhamento humano.
Feedback	Tickets e manifestações.

AtendentePro

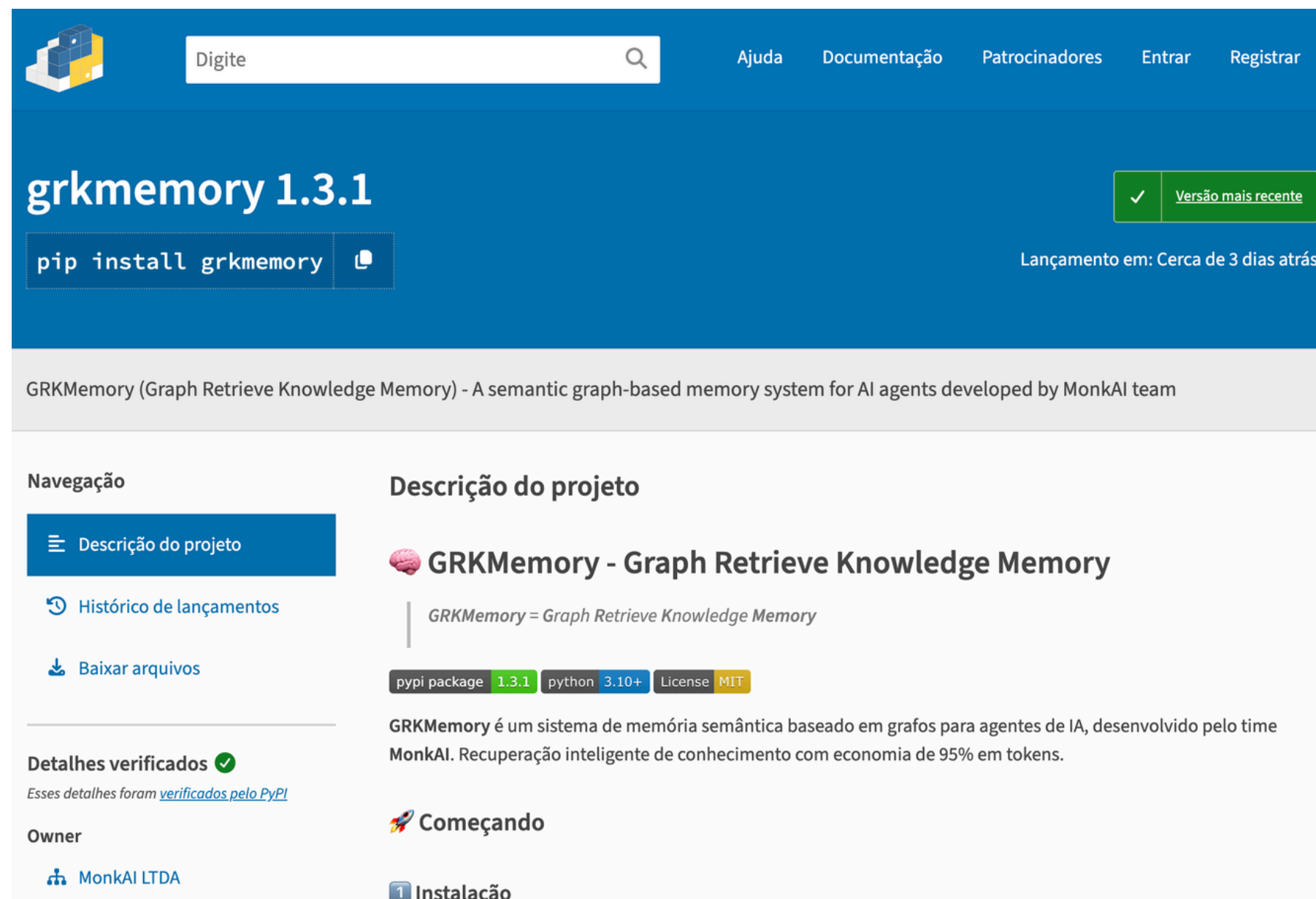
O AtendentePro combina uma biblioteca de templates, YAMLs, guardrails e agentes especializados para criar redes conversacionais padronizadas e seguras. Com isso, a aplicação consegue ser personalizada por cliente sem perder consistência operacional, escalabilidade e controle da experiência entregue ao usuário.



GRKMemory

Um sistema de memória semântica baseado em grafos para agentes de IA, desenvolvido pelo time MonkAI.

Recuperação inteligente de conhecimento com economia de 95% em tokens.



The screenshot shows the PyPI project page for grkmemory 1.3.1. The page has a blue header with a search bar and navigation links: Ajuda, Documentação, Patrocinadores, Entrar, and Registrar. The main content area is white and features the project name 'grkmemory 1.3.1' in large blue text. Below the name is a button with the command 'pip install grkmemory' and a copy icon. To the right, there is a green checkmark icon and the text 'Versão mais recente'. Below this, it says 'Lançamento em: Cerca de 3 dias atrás'. The description of the project is 'GRKMemory (Graph Retrieve Knowledge Memory) - A semantic graph-based memory system for AI agents developed by MonkAI team'. The left sidebar contains a 'Navegação' section with links for 'Descrição do projeto', 'Histórico de lançamentos', and 'Baixar arquivos'. Below that is a 'Detalhes verificados' section with a green checkmark and the text 'Esses detalhes foram verificados pelo PyPI'. The 'Owner' section shows 'MonkAI LTDA'. The main content area also has a 'Descrição do projeto' section with a brain icon and the title 'GRKMemory - Graph Retrieve Knowledge Memory'. Below the title is the text 'GRKMemory = Graph Retrieve Knowledge Memory' and a metadata bar showing 'pypi package 1.3.1 python 3.10+ License MIT'. The description text is 'GRKMemory é um sistema de memória semântica baseado em grafos para agentes de IA, desenvolvido pelo time MonkAI. Recuperação inteligente de conhecimento com economia de 95% em tokens.' Below the description is a 'Começando' section with a rocket icon and a link to 'Instalação'.

<https://pypi.org/project/grkmemory/>

GRKMemory

Exemplos como lidamos com o histórico das sessões, transformando em memórias estruturadas.

Estrutura do JSON

```
{
  "sessoes": [
    {
      "id": "sess_abc123",
      "timestamp": "2025-01-09T12:00:00",
      "summary": "Discussão sobre Python e IA",
      "tags": ["python", "ia", "programação"],
      "entities": ["Python", "OpenAI", "GPT"],
      "concepts": ["machine learning", "api"],
      "messages": [
        {"role": "user", "content": "..."},
        {"role": "assistant", "content": "..."}
      ]
    }
  ]
}
```

Estrutura do TOON (Token-Optimized Object Notation)

O mesmo conteúdo em TOON ocupa ~25% menos tokens, ideal para contexto de LLM:

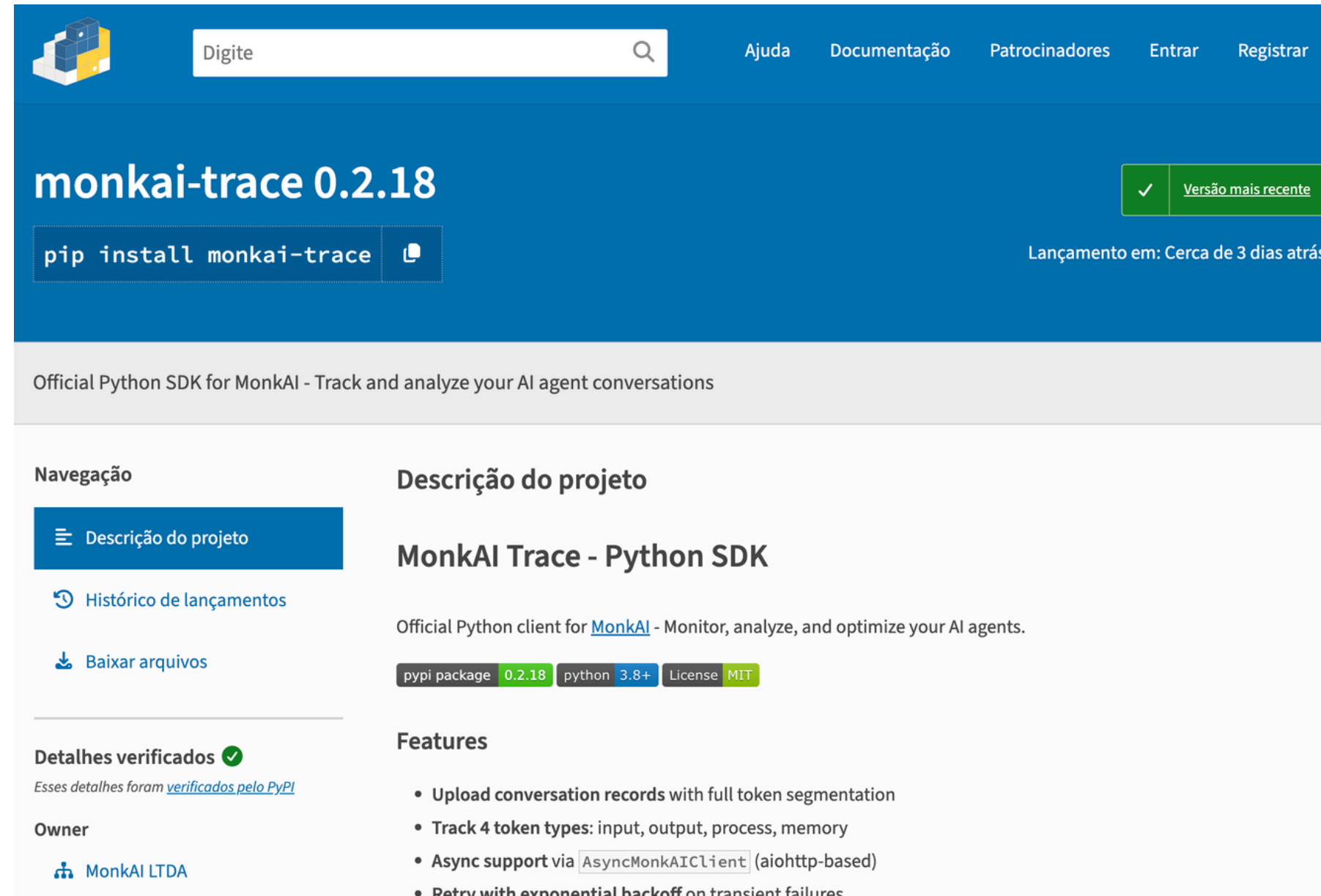
```
sessoes[1]:
- id: sess_abc123
  timestamp: "2025-01-09T12:00:00"
  summary: Discussão sobre Python e IA
  tags[3]: python,ia,programação
  entities[3]: Python,OpenAI,GPT
  concepts[2]: machine learning,api
  messages[2]{role,content}:
    user,Vamos falar sobre Python
    assistant,Claro! O que você quer saber?
```

Nota: TOON elimina chaves, colchetes e aspas redundantes, compactando listas e tabelas em notação posicional. Instale com `pip install toon_format`.

<https://pypi.org/project/grkmemory/>

MonkAI Trace

Cliente Python oficial para MonkAI – Monitore, analise e otimize seus agentes de IA.



The screenshot shows the PyPI project page for **monkai-trace 0.2.18**. The page features a blue header with a search bar and navigation links: Ajuda, Documentação, Patrocinadores, Entrar, and Registrar. Below the header, the project name and version are displayed, along with a green checkmark and the text "Versão mais recente". A button for installation is shown with the command `pip install monkai-trace`. The release date is noted as "Lançamento em: Cerca de 3 dias atrás".

Official Python SDK for MonkAI - Track and analyze your AI agent conversations

Navegação

- Descrição do projeto
- Histórico de lançamentos
- Baixar arquivos

Detalhes verificados ✓
Esses detalhes foram [verificados pelo PyPI](#)

Owner

- MonkAI LTDA

Descrição do projeto

MonkAI Trace - Python SDK

Official Python client for [MonkAI](#) - Monitor, analyze, and optimize your AI agents.

`pip package` `0.2.18` `python 3.8+` `License MIT`

Features

- Upload conversation records with full token segmentation
- Track 4 token types: input, output, process, memory
- Async support via `AsyncMonkAIClient` (aiohttp-based)
- Retry with exponential backoff on transient failures

<https://pypi.org/project/monkai-trace/>

Certificados de Qualidade

Realizamos auditorias de qualidade em nossos produtos da MonkAI, avaliando cada um deles quanto à **segurança, usabilidade e escalabilidade**. Emitido pelo Claude AI Quality Assurance Opus 4.6, alcançamos a nota **A** em todos os nossos produtos liberados.

AUTENTICIDADE — SERIAL NUMBER

MONKAI-AP-0626-F59FC5E769C0CE46

Hash de verificação SHA-256 (primeiros 16 chars):

```
echo -n "AtendentePro-v0.6.26-2026-03-01-BeMonkAI-SecurityUsabilityScalability-R5" | shasum -a 256 # Esperado: f59fc5e769c0ce46...
```

Emissor: Claude AI Quality Assurance

Data de emissão: 01 de março de 2026

Revisão: R5 (revalidação independente)

Validade: Versão 0.6.26

Este certificado é válido exclusivamente para a versão indicada. Versões subsequentes requerem nova auditoria. O uso em produção deve seguir o checklist de PRODUCTION_SECURITY.md.

✓ CERTIFICADO VÁLIDO

Claude AI · Quality Assurance · 2026

<https://www.monkai.com.br/certificados>

PRODUTOS

Distribuição



DeepRead SaaS

TrackFuel

AIConsulta

MonkaiDrop

MonkAI Hub



MonkAI Trace



GrkMemory



AtendentePro



DeepRead



CUSTO, QUALIDADE E VELOCIDADE:
COMO ESCOLHEMOS O MODELO CERTO

USE CASE - GIGI

Evolução de Acurácia — Gigi (White Martins)

gpt-4.1 | 230 interações | 14 iterações | 25/03 a 02/04



White Martins (Gigi) — Benchmark de Modelos AI v3

Data: 2026-04-02 a 2026-04-05 | Bateria: 230 interacoes, 72 grupos de teste

Modelos testados (**23**):

- **gpt-4.1,**
- **gpt-4.1-mini,**
- **gpt-5,**
- **gpt-5.1,**
- **gpt-5-mini,**
- **gpt-5.4-mini,**
- **gpt-5.4-nano,**
- **claude-sonnet-4-6,**
- **claude-opus-4-6,**
- **claude-haiku-4-5,**
- **gemini-2.5-pro,**
- **gemini-2.5-flash,**
- **grok-3,**
- **grok-3-mini,**
- **grok-4-fast-reasoning,**
- **grok-4-fast-non-reasoning,**
- **grok-4-1-fast-non-reasoning,**
- **Kimi-K2.5,**
- **Mistral-Large-3,**
- **DeepSeek-V3.2,**
- **Llama-4-Maverick,**
- **DeepSeek-R1-0528,**
- **cohere-command-a**

White Martins (Gigi) — Benchmark de Modelos AI v3

Custo x Acuracidade x Latência



Grupos de Performance

Modelo	Pass Rate	Latencia	Sender Fail	Content Fail	Observacao
gemini-2.5-pro	96.5%	11.5s	2	8	Melhor absoluto, latencia alta
gpt-4.1	92.6%	5.0s	15	4	Baseline solido, melhor instruction following
claude-sonnet-4-6	92.6%	4.1s	14	14	Empatado com gpt-4.1, mais rapido
claude-opus-4-6	92.2%	6.7s	16	10	Premium sem ganho sobre sonnet
grok-4-fast-reasoning	91.7%	8.1s	15	14	Excelente custo-beneficio
gemini-2.5-flash	89.6%	3.7s	17	23	Mais rapido dos lideres
grok-3	89.1%	6.6s	23	20	Solido mas caro

Group Pass Rate (grupo completo sem falhas)

Modelo	Interacoes Pass	Grupos Completos	Delta
gemini-2.5-pro	96.5%	88.9% (64/72)	-7.6pp
gpt-4.1	92.6%	83.3% (60/72)	-9.3pp
claude-sonnet-4-6	92.6%	83.3% (60/72)	-9.3pp
grok-4-fast-reasoning	91.7%	80.6% (58/72)	-11.1pp
gemini-2.5-flash	89.6%	77.8% (56/72)	-11.8pp
gpt-4.1-mini	80.4%	72.2% (52/72)	-8.2pp
gpt-5.4-nano	84.4%	68.1% (49/72)	-16.3pp

Testes mais difíceis (falham em múltiplos modelos líderes)

Test ID	Categoria	Falhas (de 8 modelos)	Observacao
205	Multi ServLig+AtivoOp+Indust	8/8	Fluxo mais complexo — todos falham em algum ponto
204	Multi Consumo+Comerc+Energia	8/8	Troca de contexto entre 3 categorias
201	Multi Comerc+Frete+Consumo	8/8	Frete (F1) e o IVA mais errado
203	Multi AtivoPj+Indust+Frete	8/8	Frete (F5) confunde todos os modelos
202	Multi AtivoOp+Energia+ServNaoLig	6/8	gemini-pro e grok-4-reasoning acertam
121	Usage Plataforma	8/8	Provavel problema no teste
200	Multi Energia+Indust+ServOp	5/8	Menos complexo, mais acertos

Recomendações

1. **Avaliar migração para gemini-2.5-pro** — 96.5% supera todos os modelos, custo razoável (\$2.70/benchmark)
2. **Avaliar grok-4-fast-reasoning** — 91.7% a \$0.38/benchmark, 10x mais barato que gpt-4.1
3. **Considerar gpt-5.4-nano para reduzir custo** — 84.4% a \$0.39/benchmark, latência de 3.9s
4. **Manter gpt-4.1 como fallback** — mais testado e estável em produção
5. **Não usar claude-opus-4-6** — mesmo resultado que sonnet-4-6 por quase 2x o custo
6. **Não usar família gpt-5.0/5.1 para multi-agent** — instruction following insuficiente
7. **Não investir em variantes non-reasoning** — perda de 28pp vs reasoning

MonkAI - Deep tech

A MonkAI é uma deep tech porque desenvolve tecnologia de base — de memória conversacional a frameworks de agentes — com pesquisa proprietária, eficiência comprovada e foco em infraestrutura e soberania dos dados.

MonkAI vs ChatGPT: Avaliação Comparativa de Desempenho, Confiabilidade e Escalabilidade no Processamento Automatizado de Documentos Corporativos*

Arthur Vaz, AND José Riveaux, AND Davi Leal,

arthur.vaz@monkai.com.br jose.riveaux@monkai.com.br davi.leal@monkai.com.br

MonkAI Team

Este artigo apresenta um estudo experimental que compara o pipeline proprietário *MonkAI* ao *ChatGPT-4o* no processamento automatizado de documentos corporativos. Em 36 arquivos reais (artigos, divulgação de resultados, contratos e editais) submetemos cada modelo a três tipos de consulta: *simples* (campo pontual), *agregadora* (síntese de múltiplos trechos) e *interpretativa* (inferência contextual). Medições de latência, confiabilidade e escalabilidade mostram que o MonkAI mantém ~96% de respostas confiáveis com latência linear previsível, ao passo que o ChatGPT apresenta maior variabilidade e queda de desempenho fora do horário comercial. Para fluxos 24x7 sob SLAs rígidos, o MonkAI reduz risco operacional e oferece transparência de custos, enquanto o ChatGPT pode atuar como acelerador oportunístico. Trabalhos futuros incluirão testes em escala de produção, expansão do corpus e métricas de sustentabilidade energética.

Keywords— Document automation, Latency benchmark, Confidence metrics, Large Language Models

TOON vs JSON: Evaluating Token-Oriented Serialization for LLM Memory Systems

Arthur Vaz, Monkai Research Team
MonkAI - Graph Retrieve Knowledge Memory
Rio de Janeiro, Brazil
contato@monkai.com.br

Abstract—Large Language Model (LLM) applications increasingly rely on persistent memory systems to maintain context across conversations. The serialization format used for storing and transmitting this context directly impacts both computational costs and token consumption. We present a comprehensive evaluation comparing JSON (JavaScript Object Notation) with TOON (Token-Oriented Object Notation), a compact schema-aware format designed specifically for LLM workloads. Through extensive benchmarking on datasets ranging from 10,000 to 100,000 memory sessions, we demonstrate that TOON achieves 20–28% reduction in token consumption when transmitting context to LLMs, while JSON maintains 27–64× faster parsing performance due to native runtime support. We propose a hybrid architecture that leverages JSON for persistent storage and real-time retrieval, converting to TOON only at the LLM interface boundary. This approach preserves sub-millisecond retrieval latency while reducing inference costs by approximately 25%. Additionally, we introduce a multi-criteria retrieval algorithm that considers seven weighted characteristics simultaneously, achieving 100% accuracy in finding specific information across 100,000 sessions in under 3 milliseconds.

Index Terms—serialization formats, token optimization, conversational memory, LLM efficiency, TOON, JSON, multi-

processing overhead in the context of conversational memory systems.

Our contributions are threefold:

- 1) A comprehensive benchmark comparing JSON and TOON across storage size, parsing speed, and token efficiency for memory workloads ranging from 10,000 to 100,000 sessions.
- 2) A multi-criteria retrieval algorithm that evaluates seven weighted characteristics simultaneously, enabling precise memory selection in large-scale deployments.
- 3) A hybrid architecture recommendation that optimizes for both retrieval latency and LLM token consumption.

II. BACKGROUND

A. Token Economics in LLM Applications

Modern LLM APIs charge based on token consumption, with input tokens typically priced at 50–75% of output tokens. For memory-augmented applications, retrieved context can

Newer Is Not Always Better: A 23-Model Benchmark Reveals That Infrastructure and Instruction Fidelity Outweigh Scale in Multi-Agent LLM Orchestration*

MonkAI Research Team
contato@monkai.com.br

MonkAI

Deploying large language models (LLMs) as orchestrators in multi-agent customer-service pipelines requires reliable instruction following, correct inter-agent routing, and predictable latency—properties that single-turn benchmarks fail to capture. We present a controlled evaluation of 23 commercial LLMs across 230 interactions grouped into 72 conversational test suites, conducted on a production-grade multi-agent system for an industrial-gas distributor. Models were assessed on pass rate, agent-routing accuracy, per-category performance, latency, and estimated cost. Gemini 2.5 Pro achieved the highest pass rate (96.5%), while Grok-4-Fast-Reasoning delivered the best cost-performance ratio (91.7% at \$0.38 per benchmark run). We find that (i) multi-step context switching is the strongest discriminator of model quality, (ii) the choice of inference API can shift accuracy by up to 83 percentage points for the same model, and (iii) chain-of-thought reasoning yields a 28 p.p. gain over non-reasoning variants when properly bounded. These results provide actionable guidance for practitioners selecting backbone LLMs for agentic workflows.

Keywords—Large language models, multi-agent systems, benchmark, customer service, tool use, instruction following

0 Introduction

Multi-agent architectures, in which a central orchestrator delegates sub-tasks to specialised agents via structured handoffs, have emerged as the dominant pattern for deploying LLMs in enterprise workflows [1, 2]. Unlike single-turn question answering, these systems require the back-

system deployed for an industrial-gas company. Our contributions are threefold:

1. A benchmark of 230 interactions across 14 business categories that measures both per-interaction and per-conversation (group-complete) pass rates.
2. A systematic analysis of routing failures, revealing

Reducing Retrieval Cost and Carbon Footprint with Graph-Aware Conversational Memory

Monkai Research Team
MonkAI
Rio de Janeiro, Brazil
contato@monkai.com.br

Abstract—Large Language Models (LLMs) are increasingly embedded in customer-facing workflows, yet most deployments still serialize entire conversational histories into prompts. This strategy inflates token consumption, induces latency, and increases the carbon footprint of AI services. We present Graph Retrieve Knowledge Memory (GRKM), a graph-aware retrieval layer that learns high-salience relationships between dialog sessions and enforces selective context delivery. While the internal scoring pipeline is proprietary, we summarize the methodology and report three findings across deployments that process up to 20,000 requests per month: (i) prompt tokens fall by 75% on average, lowering monthly spend from \$1,800 to \$440, (ii) response latency drops from 3.2s to 1.1s, and (iii) energy usage decreases by 61%, yielding measurable environmental gains. We also discuss qualitative improvements in answer accuracy obtained through selective replay of car-related conversations, showing that the graph representation faithfully retrieves prior discussions without manual curation.

Index Terms—retrieval-augmented generation, conversational AI, knowledge graphs, energy-aware inference

semantic attributes—tags, named entities, and key points—combined with cosine similarity between LLM-generated embeddings. A densification procedure assigns weights to edges according to (i) overlap of extracted attributes and (ii) embedding similarity above a configurable threshold (0.5 in our deployments). Node density corresponds to the mean weight of incident edges, highlighting sessions that act as hubs for frequently revisited topics.

A. Query Profiling

Each user message is transformed using the same summarization pipeline that structures our knowledge base. The “query profile” produces normalized tags, entities, and key points, plus an embedding computed via OpenAI text-embedding-3-large. By mirroring the session representation, we obtain lexical and semantic features that align with the graph’s attribute space without hand-crafted dictionaries.

Generic vs Per-Provider Prompts in Multi-Agent LLM Pipelines*

MonkAI Research Team
contato@monkai.com.br

MonkAI

Practitioners deploying large language models (LLMs) behind multi-agent pipelines face a recurring question: does tailoring the system prompt to each backbone model improve accuracy, or is a single generic prompt sufficient? We study this question through a controlled ablation on a production industrial-gas customer-service pipeline comprising eight specialised agents connected by structured handoffs. The corpus consists of 230 paired interactions per model across 72 multi-step conversational flows, split 70/30 into training (165 interactions) and held-out test (65 interactions) subsets. Four backbone models are evaluated: Claude Sonnet 4.6, Gemini 2.5 Flash, Grok-4-Fast-Reasoning, Grok-4-Fast-Non-Reasoning. The ablation compares three customisation strategies against a shared-prompt baseline: (i) *rule-based dialect adaptation* (per-family XML/markdown/stripped-parenthesis transforms), (ii) *instruction-only optimisation* via APE [1] and MIPROv2 [2], and (iii) *few-shot demo optimisation* via DSPy Bootstrapping/FewShot [3]. On the held-out test set under a deduction-based graded rubric, rule-based dialect adaptation produces a statistically significant regression on Gemini ($\Delta = -0.32$ graded score, Wilcoxon $p = 0.032$) and no gains on the five other models tested; instruction-only optimisation yields null results on all four DSPy-targeted models; few-shot demos yield the only positive signal— $\Delta = +0.108$ boot-derived accuracy on Gemini (McNemar $p = 0.016$, survives Bonferroni correction at 3 tests) and a directionally-consistent but uncorrected-marginal $\Delta = +0.107$ on Grok-Non-Reasoning ($p = 0.092$). We frame the empirical pattern—only the strategy that provides examples yields a measurable gain—as a *working hypothesis* rather than a general law, given that the positive signal rests on one robust and one marginal observation, relies on a single LLM judge, and may reflect a statistical-power asymmetry between optimisers as well as a genuine methodological one. Methodological contributions include the silent-exception fallback bug that invalidated our earlier findings once caught, a deduction-based rubric that escapes bimodal LLM-judge saturation, and a cache-directory scoping pattern that enables clean per-optimiser A/B evaluation.

Keywords— Prompt optimisation, multi-agent systems, DSPy, automatic prompt engineering, few-shot learning, LLM-as-judge evaluation

Nossos usuários



**Unlock a New Horizon of
Productivity, Be Monkai.**

WWW.BEMONKAI.AI
ARTHUR LAMBLET VAZ

