



luciano-coelho / OpenInsuranceAgent

[Code](#)[Issues](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)

OpenInsuranceAgent



/ README.md

in [main](#)[Cancel changes](#)[Commit changes...](#)[Edit](#)[Preview](#) Show Diff

Open Insurance Agent

Um agente de IA modular e auditável para análise normativa do Open Insurance Brasil

Visão Geral

O Open Insurance Agent é um projeto de pesquisa aplicada desenvolvido por Luciano Coelho, doutorando em Ciência da Computação pela UFSC, vinculado ao Laboratório de Segurança em Computação (LabSEC), sob orientação do Prof. Ricardo Custódio.

A solução combina técnicas de RAG (Retrieval-Augmented Generation) e embeddings regulatórios com modelos de linguagem auditáveis, oferecendo um protótipo inovador para análise automatizada e segura de normas do Open Insurance Brasil. Essa abordagem permite rastreabilidade das fontes, auditoria das respostas e avaliação contínua de métricas de precisão e aderência normativa, garantindo maior confiabilidade nas interpretações automatizadas de documentos oficiais.

A iniciativa integra esforços de inovação regulatória, inteligência artificial e cibersegurança no contexto do Open Insurance Brasil (OPIN) — programa supervisionado pela Superintendência de Seguros Privados (SUSEP) e vinculado ao ecossistema Open Finance Brasil.

O projeto propõe um agente inteligente especializado em normas, diretrizes e padrões técnicos do Open Insurance, capaz de:

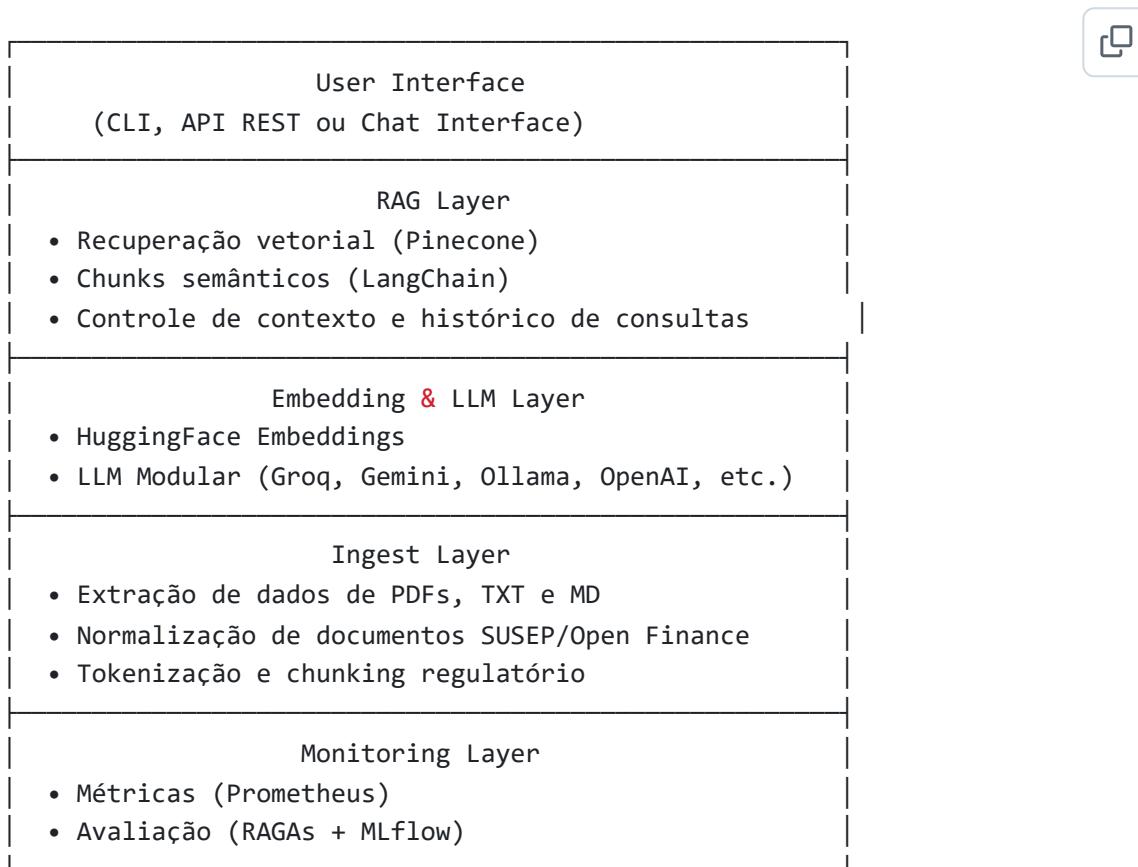
- interpretar documentos oficiais da SUSEP, CNSP e CMN;

- responder a consultas técnicas e regulatórias de forma fundamentada;
- gerar análises rastreáveis, explicáveis e auditáveis.

Arquitetura do Sistema

O sistema é baseado em uma **arquitetura RAG (Retrieval-Augmented Generation)**, permitindo que as respostas da IA sejam sempre **fundamentadas em documentos oficiais**.

O fluxo completo é dividido em cinco camadas:



Características Principais

1. Modularidade da IA

O agente não está vinculado a uma IA específica. Ele opera sob um modelo de acoplamento dinâmico, aceitando qualquer provedor de modelo de linguagem (LLM) que possua API Key válida e suporte ao endpoint compatível com o padrão OpenAI-like.

Atualmente, são suportados:

- Groq API (mixtral, llama3, gemma2);
- Google Gemini (vía Vertex ou REST);
- OpenAI (gpt-4o, gpt-4-turbo);
- Ollama (modelos llama3, mistral, phi3);

Essa modularidade garante portabilidade, redundância e independência tecnológica, permitindo continuidade operacional mesmo diante de descontinuações de modelos.

2. Base Regulamentar

Os documentos utilizados são fontes oficiais, públicas e auditáveis, incluindo:

- Circulares, Resoluções e RDDs da SUSEP;
- Documentos técnicos da OPIN (Diretório de Participantes, Certificação, Glossário, etc.);
- Manuais e guias publicados no Portal da SUSEP e no repositório público do Open Insurance Brasil.

Esses materiais são armazenados localmente em `/data/oi` e ingeridos via `scripts/ingest_local.py`.

3. Ingestão e Indexação

A pipeline de ingestão realiza:

- Extração e normalização de documentos (PyPDFLoader, UnstructuredMarkdownLoader);
- Divisão em blocos semânticos (chunks) configuráveis via .env;
- Criação e sincronização de embeddings (all-MiniLM-L6-v2) no Pinecone;
- Monitoramento de volume e latência para cada ciclo de ingestão.

O resultado é um índice vetorial consistente, capaz de responder a consultas com precisão contextual.

4. Consultas Inteligentes

As consultas podem ser realizadas via CLI:

python -m scripts.ask_oi "Quais são os requisitos de certificados para c [Copiar]

O agente realiza:

- Recuperação dos 5 trechos mais relevantes (Top-K);
- Consolidação do contexto em prompt estruturado;
- Geração de resposta fundamentada e rastreável, com metadados da fonte.

5. Base Conceitual e Pesquisa

O projeto está ancorado em princípios acadêmicos e regulatórios sólidos:

"A solução proposta combina técnicas de RAG e embeddings regulatórios co [Copiar]

Essa estrutura visa:

- Reduzir erros humanos em consultas normativas;
- Acelerar a conformidade regulatória;
- Garantir rastreabilidade das respostas e explicabilidade algorítmica;
- Fortalecer a interoperabilidade entre Open Finance e Open Insurance;
- Apoiar a SUSEP e instituições participantes na supervisão e implementação de fases do Opin.

6. Riscos e Cuidados

- O sistema deve ser constantemente atualizado para refletir mudanças regulatórias.
- É essencial manter mecanismos de auditoria e logging completo.
- As conexões externas (Pinecone, LLM APIs) devem seguir padrões de criptografia TLS 1.3.

O uso institucional requer validação prévia da SUSEP e acompanhamento técnico de conformidade.

7. Métricas e Observabilidade

O agente coleta métricas em tempo real via Prometheus, permitindo monitorar:

- Latência média de consulta;
- Taxa de acerto semântico (via RAGAs);
- Consumo de tokens e custo operacional;
- Disponibilidade e tempo de resposta das APIs conectadas.

Os experimentos são versionados via MLflow e avaliados sob metodologia A/B com diferentes LLMs.

Configuração do Ambiente

1. Pré-requisitos

Certifique-se de ter instalado:

- Python 3.10+
- Git
- Virtualenv (recomendado)
- Conta ativa no [Pinecone](#)
- API Key válida para o provedor de LLM (Groq, Gemini, OpenAI, etc.)

2. Clonar o repositório

```
git clone https://github.com/luciano-coelho/OpenInsuranceAgent.git  
cd open-insurance-agent
```



3. Criar e ativar o ambiente virtual

```
python -m venv .venv  
# Ativar o ambiente  
  
# Windows  
.venv\Scripts\activate  
  
# Linux/Mac  
source .venv/bin/activate
```



4. Instalar dependências

```
pip install --upgrade pip  
pip install -r requirements.txt
```



5. Configurar variáveis de ambiente

Crie um arquivo .env na raiz do projeto com o seguinte conteúdo:

```
# ---- LLM / Embeddings ----  
GROQ_API_KEY=your_groq_key  
GEN_MODEL=llama3-8b-8192  
EMBED_MODEL=all-MiniLM-L6-v2  
  
# ---- Pinecone ----  
PINECONE_API_KEY=your_pinecone_key  
PINECONE_ENVIRONMENT=us-east-1  
PINECONE_INDEX_NAME=open-insurance-index  
  
# ---- RAG ----  
TOP_K=5  
CHUNK_SIZE=600  
CHUNK_OVERLAP=80
```



⚠️ Observação: O agente é modular — ele não está vinculado a uma IA específica. Basta trocar a chave e o nome do modelo no .env para usar Groq, Gemini, OpenAI, Ollama ou qualquer outro LLM compatível com API REST no padrão OpenAI-like.

6. Adicionar os documentos oficiais

Coloque os arquivos normativos e técnicos em:

/data/oi/



Supora os formatos:

- .pdf
- .txt
- .md

Exemplo: Circulares SUSEP, Resoluções CNSP, RDDs, perfis de segurança do Open Finance, etc.

7. Executar a ingestão de documentos

```
python -m scripts.ingest_local
```



Este comando irá:

- Ler e processar os documentos;
- Gerar chunks semânticos;
- Criar embeddings e enviar para o Pinecone.

8. Verificar o status do índice

```
python -m scripts.check_pinecone
```



Exibe o total de vetores, status do índice e amostra de metadados armazenados.

9. Realizar consultas

```
python -m scripts.ask_oi "Quais são os requisitos de certificados para c
```



Autores e Colaboradores

Luciano Coelho — Doutorando em Ciência da Computação (UFSC / LabSEC)

Prof. Ricardo Custódio — Orientador (UFSC / LabSEC)

Manuel Matos — Coautor (Câmara e-net)

Drª. Patricia Figueiredo - Seleção de documentos normativos (Open Power Corretora de Seguros SA)