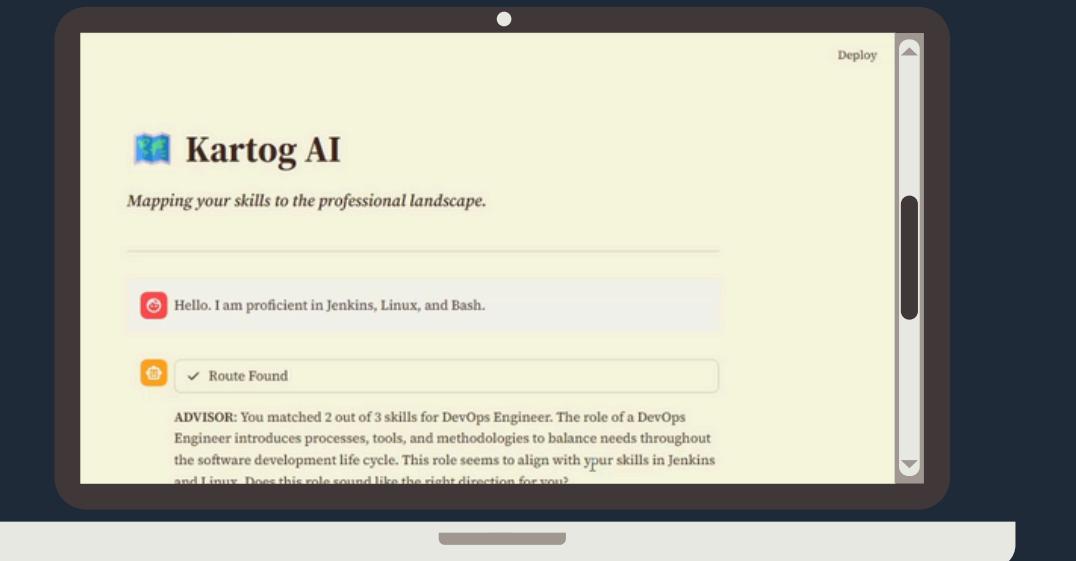
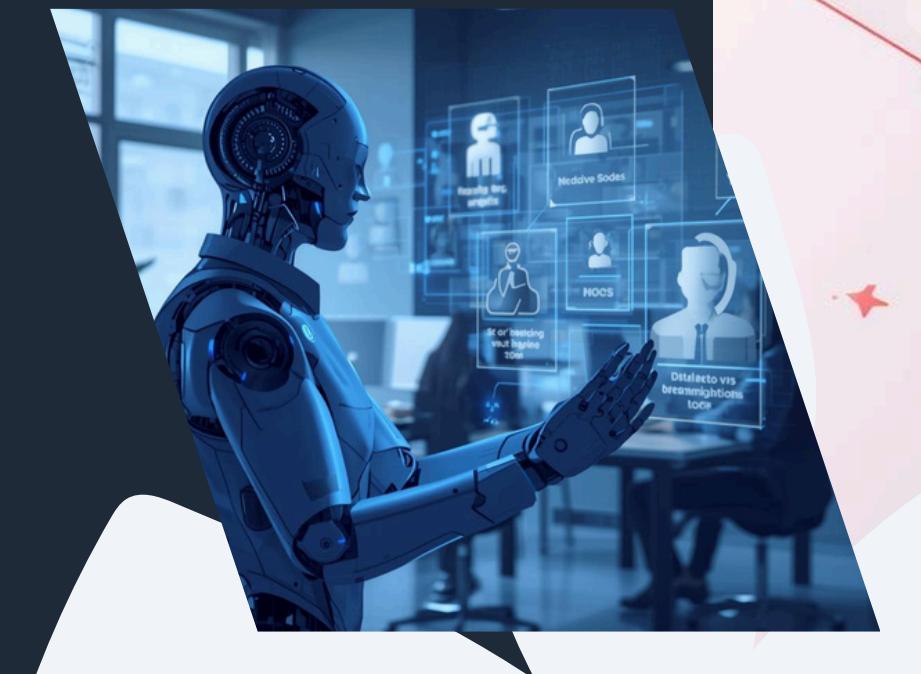


Kartog AI – Career Intelligence Platform

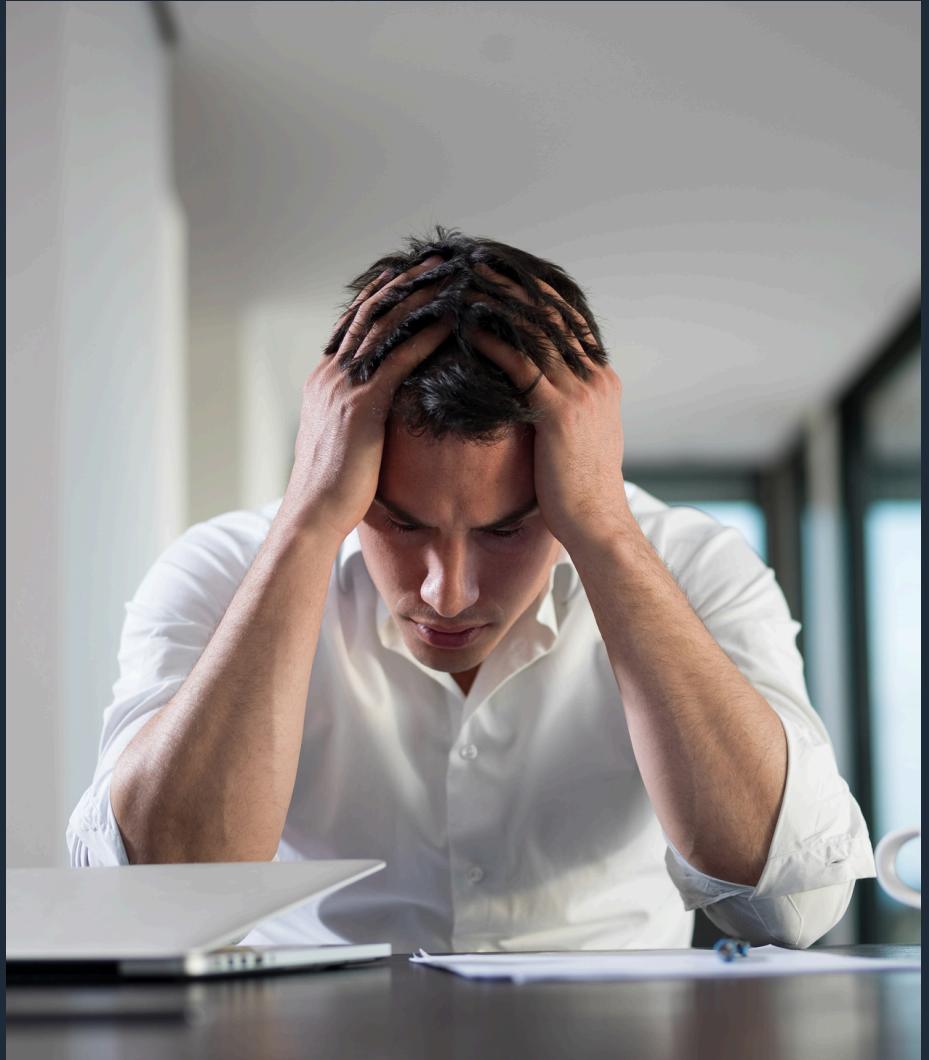
ANTONIO GRIGORASCU & DANIEL BIRSAN



Caching.

Introducere

PROBLEM STATEMENT



Cel puțin o dată în viață, fiecare dintre noi se confruntă cu dificultatea căutării unui loc de muncă, un proces adesea repetitiv și inefficient, în care utilizatorii sunt nevoiți să își introducă aceleași informații pe multiple platforme și să se bazeze pe algoritmi care funcționează, în majoritatea cazurilor, pe principiul keyword matching, potrivirea exactă a cuvintelor. Această abordare ignoră nuanțele semantice ale competențelor (de exemplu, faptul că „Data Science” implică utilizarea limbajului Python chiar dacă nu este menționat explicit) și, mai important, nu oferă utilizatorului niciun context sau ghidaj, rezultând într-o listă de joburi fără explicații clare privind nepotrivirile sau pașii necesari pentru progresul profesional.

Solutia propusă

KARTOG AI ESTE O PLATFORMĂ SOFTWARE PENTRU ORIENTARE PROFESSIONALĂ INTELIGENTĂ,
BAZATĂ PE:

- agenți AI coordonați prin LangGraph
- ontologie de cariere (Neo4j)
- date reale din piața muncii (MongoDB)
- caching semantic pentru performanță și scalabilitate



SCOPUL ESTE TRANSFORMAREA UNEI SIMPLE CĂUTĂRI DE JOBURI ÎNTR-UN PROCES DE
CONSULTANȚĂ DE CARIERĂ.



Yes, this looks like the right opportunity for me.



SCOUT: To find active job listings for a DevOps Engineer, I need to know your location and experience level. Can you please provide me with your location and experience level?



I am in Newcastle and I am mid-level.



✓ Route Found

SCOUT: I've found some job listings for a DevOps Engineer in Newcastle. The top result is for a DevOps Engineer at DevOps Solutions with a salary range of £50,000 - £75,000. There are also some other related job listings, such as Cybersecurity Engineer and AI/ML Engineer, but they may not be an exact match for the role you're looking for. Would you like me to refine the search or provide more information about these job listings?



De ce folosim cache?

ÎN APLICAȚIILE AI MODERNE APAR URMĂTOARELE PROBLEME:

- apeluri LLM costisitoare și lente
- interogări redundante către baze de date
- lipsa reutilizării răspunsurilor similare semantic
- lipsa mecanismelor de invalidare a cache-ului



Arhitectura High-Level

COMPONENTELE PRINCIPALE:

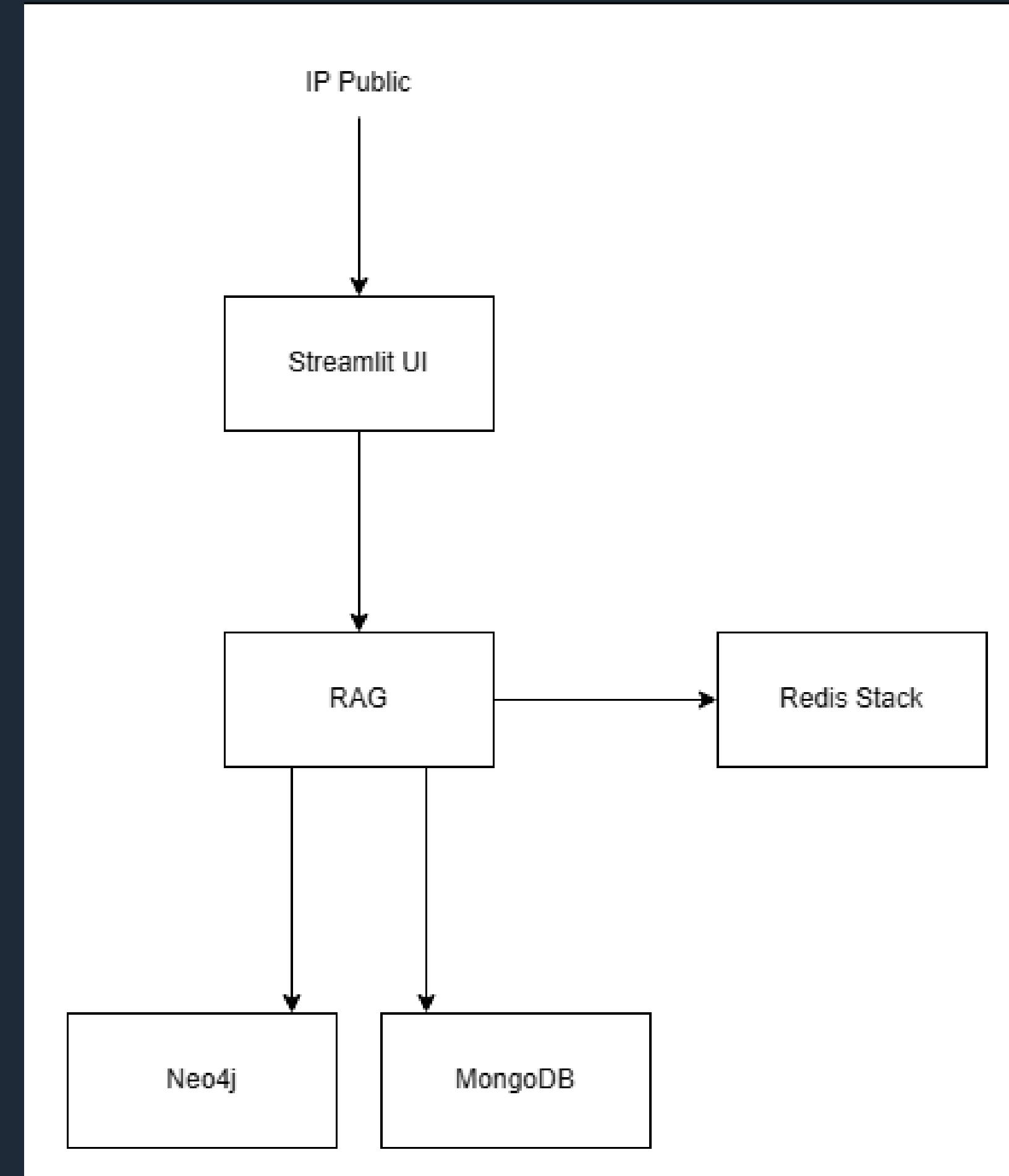
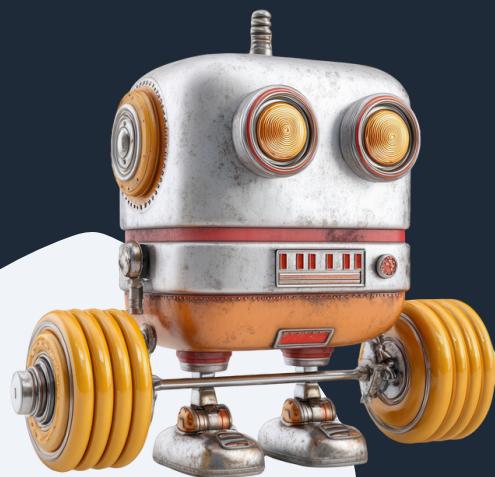


- Python App (Interfață Web + Logică AI)
- Watcher (Cache Invalidation)
- Redis Stack (Cache + Vector Store)
- Neo4j (Knowledge Graph)
- MongoDB (Document + Vector Search)
- Prometheus (Metrics)
- Grafana (Visualizare)

Toate serviciile rulează containerizat.

Execution flow

- Input utilizator (limbaj natural)
- Vectorizare input
- Căutare în Semantic Cache (Redis)
- MISS -> procesare AI
- Interogări Neo4j + MongoDB
- Răspuns final + salvare în cache



Stratul de Date:

NEO4J

Neo4j este utilizat pentru:

- modelarea ontologică a carierelor
- relații semantice între skill-uri și roluri
- interogări bazate pe structură, nu text

Avantaj față de SQL:

- relații explicite
- traversări rapide
- explainability logică



MONGODB

MongoDB este utilizat pentru:

- stocarea joburilor reale
- căutare vectorială pe descrieri
- filtrare deterministă (locație, experiență)

Abordare hibridă:

- Vector Search + Regex Filtering



De unde am luat datele?

În forma inițială, am identificat un set de date în format .xlsx, disponibil pe platforma Kaggle.

Job Title	Job Description	Required Skills	Salary Range	Location	Company	Experience Level	Industry	Job Type	Date Posted
Software Engineer	Develop software	Java, Python	£40,000 - £50,000	London	ABC Tech	Entry-Level	Technology	Full-Time	1/5/2023
Data Analyst	Analyze data	SQL, Excel	£35,000 - £45,000	Manchester	XYZ Analytics	Junior	Analytics	Full-Time	2/10/2023
Network Engineer	Maintain networks	Cisco, WAN	£45,000 - £55,000	Birmingham	Network Solutions	Mid-Level	Networking	Full-Time	3/15/2023
Cloud Architect	Design cloud	AWS, Azure	£60,000 - £70,000	Edinburgh	Cloud Innovations	Senior	Cloud Computing	Full-Time	4/20/2023
Cybersecurity Analyst	Protect data	Cybersecurity	£50,000 - £60,000	Glasgow	SecureGuard	Mid-Level	Security	Full-Time	5/25/2023
IT Project Manager	Manage projects	PMP, Agile	£70,000 - £80,000	Leeds	Project Leader	Senior	Project Management	Full-Time	6/30/2023
Data Scientist	Analyze big data	R, Python	£55,000 - £65,000	Bristol	Data Insights	Mid-Level	Data Science	Full-Time	7/5/2023

```
[{"role": "AI Ethics Consultant", "description": "Advises organizations on the ethical implications of AI development", "domain": "AI & Machine Learning", "must_have_skills": ["AI Ethics", "Data Privacy", "Risk Assessment", "Regulatory Compliance"], "nice_to_have_skills": ["Python", "GDPR", "Philosophy", "Bias Mitigation"], "skill_relationships": [{"source": "AI Ethics", "target": "Artificial Intelligence", "type": "GOVERNS"}, {"source": "Bias Mitigation", "target": "Machine Learning", "type": "IMPROVES"}, {"source": "GDPR", "target": "Data Privacy", "type": "REGULATES"}]}
```

Pentru construcția bazei de date graf (Neo4j), am utilizat Gemini Pro ca motor de generare sintetică. Modelul a preluat titlurile joburilor ca input și a mapat automat o ontologie completă, definind entitățile (nodurile) și legăturile semantice (relațiile) dintre ele.



Logica de Caching Semantic (Redis)

Proiectul utilizează o arhitectură **hibridă** de caching pentru a limita cosumul de 'tokens' al modelelor de LLM utilizate (ex: funcția `extract_skills_with_semantic_cache`) și a optimiza interacțiunea cu bazele de date Neo4j și MongoDB (ex: funcțiile `find_best_role_match` și `search_mongodb_jobs`).

Pentru implementarea mecanismului de Semantic Caching, sistemul utilizează imaginea `redis/redis-stack:latest` în detrimentul distribuției standard `redis`. Această alegere este motivată de faptul că, spre deosebire de versiunea standard, `redis-stack` include nativ modulele `RediSearch` și `RedisJSON`, transformând baza de date într-un vector store performant. Acest lucru permite stocarea interogărilor sub formă de embeddings și calcularea în timp real a distanței semantice.





Strategia de Invalidare a Cache-ului

Deoarece Semantic Caching stochează răspunsuri pentru perioade lungi (24h), există riscul ca datele servite să devină "stale" (învechite) atunci când starea reală a joburilor se modifică în baza de date principală (MongoDB).

Pentru a remedia acest lucru, sistemul implementează o arhitectură reactivă: un serviciu de tip "Watcher" ascultă modificările din MongoDB în timp real și curăță selectiv intrările relevante din Redis.

Configurare MongoDB

Pentru ca sistemul să poată invalida cache-ul în momentul ștergerii unui job, baza de date trebuie să furnizeze starea documentului înainte de ștergere (fullDocumentBeforeChange).

Acest lucru necesită activarea changeStreamPreAndPostImages pe colecția job_postings. Această comandă a fost executată în mongosh:

```
db.runCommand({  
    collMod: "job_postings",  
    changeStreamPreAndPostImages: { enabled: true }  
})
```

Fără această configurare, evenimentele de tip delete nu ar conține titlul jobului șters, făcând imposibilă invalidarea cache-ului semantic asociat.

Pentru implementare am constituit un script dedicat (watcher.py) care utilizează MongoDB Change Streams pentru a monitoriza operațiunile de scriere (CRUD).

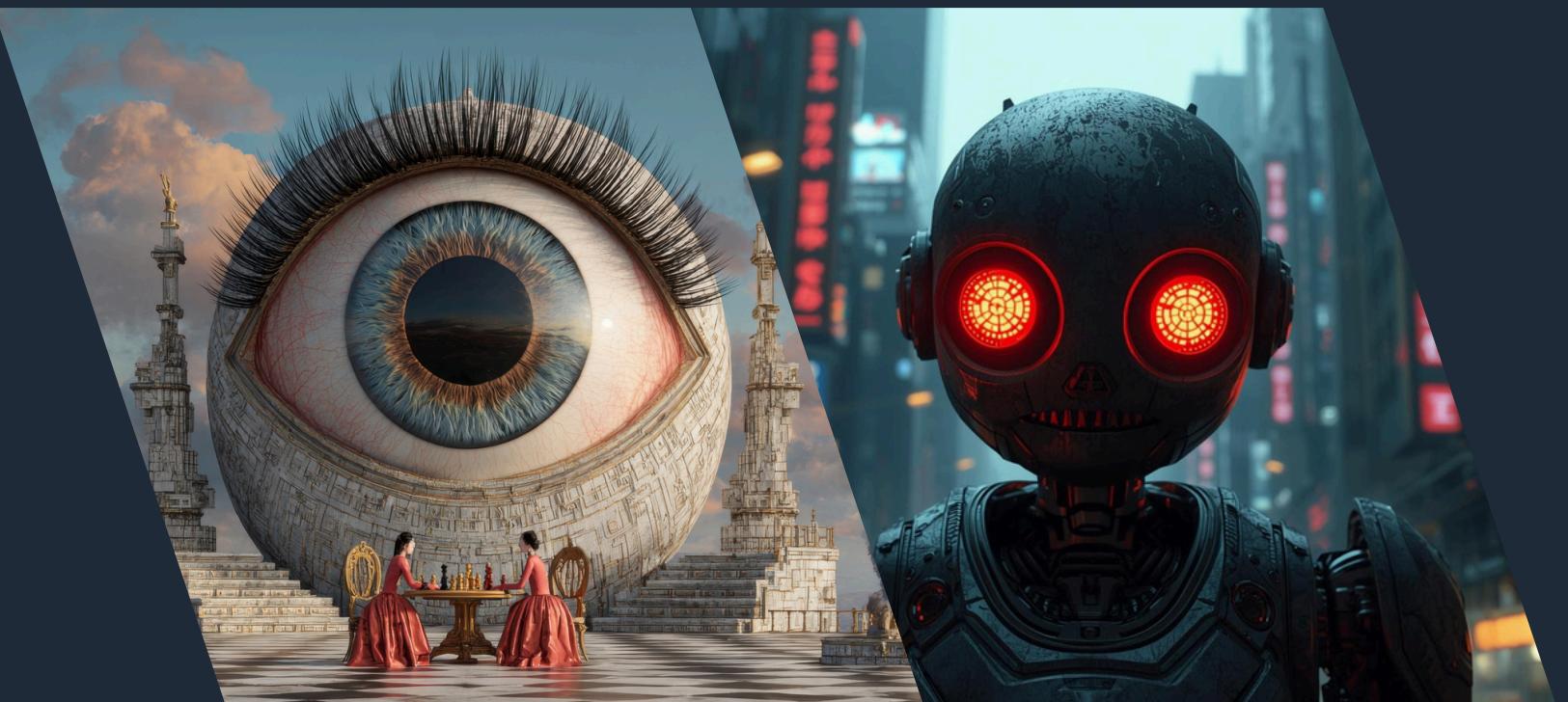
- Monitorizează operațiile CRUD asupra colecției job_postings
- Reacționează în timp real la INSERT / UPDATE / DELETE
 - Invalidare Cache Semantic (Redis):
 - Nu se șterge o cheie individuală (chei necunoscute a priori)
 - Se execută o căutare inversă în indexul semantic Redis

```
Query(f'@query_text:{term}').no_content()
```

- RediSearch returnează ID-urile documentelor cache relevante
- Intrările identificate sunt eliminate prin DELETE

```
keys_to_delete = [doc.id for doc in result.docs]

if keys_to_delete:
    redis_client.delete(*keys_to_delete)
```

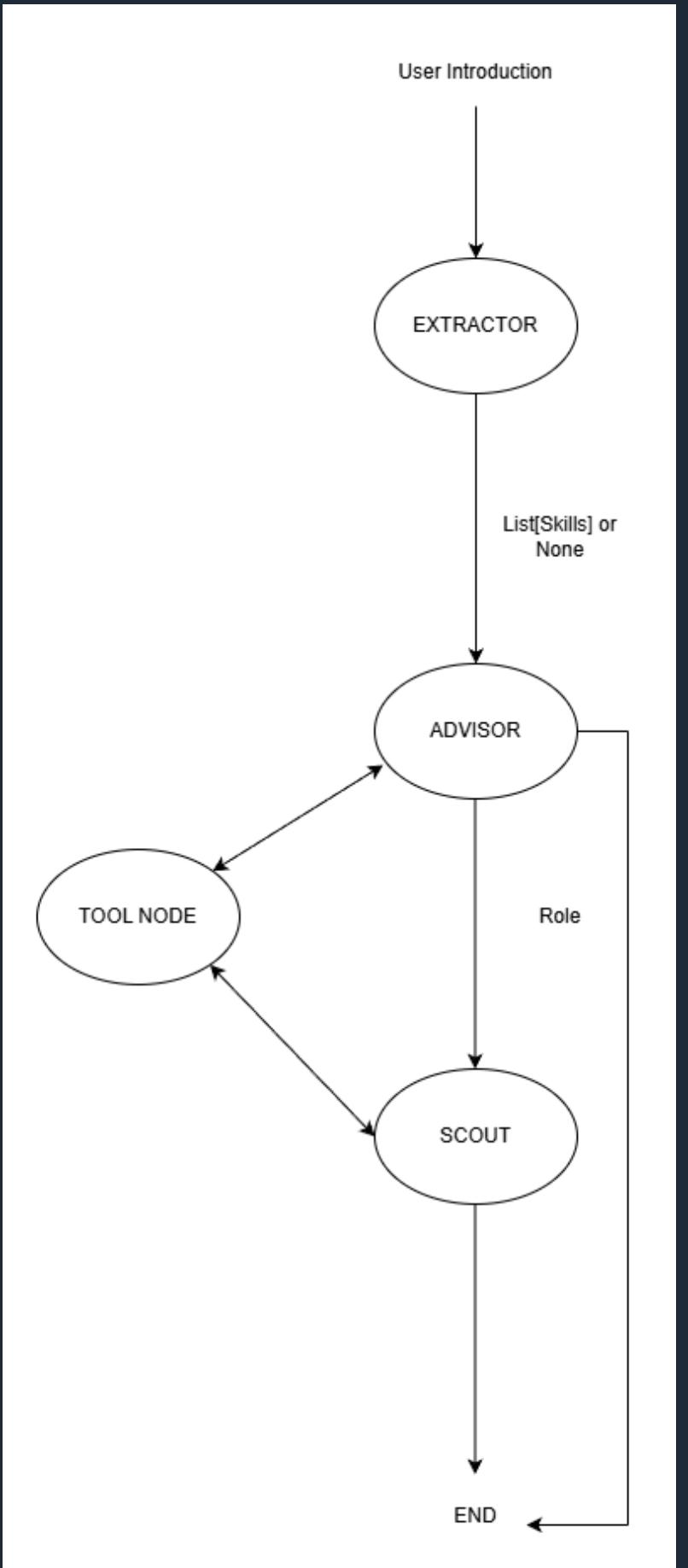


Logica AI și RAG



Workflow-ul este definit ca un graf direcționat de stări (StateGraph), unde fiecare nod reprezintă o etapă distinctă de procesare, iar muchiile ("edges") definesc logica de rutare. Sistemul menține un Global State (CareerState) care persistă datele critice (abilități, rol identificat, agent activ) pe tot parcursul conversației.

- **Nodul de Inițializare:** `run_extractor` - Nu este un agent conversațional, ci un nod de procesare "silenețios".
- `run_advisor` - se ocupă de maparea Competențe → Rol.
- `handle_tool_call` - Aceasta este un nod specializat care interceptează apelurile către funcțiile externe (**Neo4j, MongoDB**).
- `run_scout` - Aceasta este agentul final, activat doar după ce un rol a fost identificat. Preia state.`identified_role` (ex: "DevOps Engineer") și îl injectează în propriul prompt.



Logica de Rutare (Conditional Edges)

Graful utilizează funcții de rutare condițională ("routers") pentru a determina dinamic următorul pas în execuție:

- route_advisor:
 - Dacă LLM-ul cere o unealtă -> merge la nodul tools.
 - Dacă LLM-ul a emis "HANDOFF" -> merge la nodul scout.
 - Altfel -> END (Așteaptă input nou de la utilizator).
- route_scout:
 - Dacă LLM-ul cere o unealtă -> merge la nodul tools.
 - Altfel -> END (Așteaptă input nou de la utilizator).
- route_tools:
 - După executarea unei unele, fluxul se întoarce automat la agentul care a inițiat cererea (advisor sau scout), asigurând continuitatea conversației.



Tool-uri și LangChain



Arhitectura sistemului utilizează o strategie de tip "Tiered Inference" (Inferență Etajată), alocând modele diferite în funcție de complexitatea sarcinii pentru a optimiza raportul cost-performanță-viteză.

1. Modelul de Extractie (High-Speed) Utilizat pentru sarcini simple, de tip parsing (extragerea entităților din text), unde viteza este critică.

```
ChatGroq(model="llama-3.1-8b-instant", temperature=0)
```

1. Modelul de Execuție și Raționament (High-Intelligence) Utilizat pentru agenții principali (Advisor, Scout), unde sunt necesare capacitați avansate de raționament, planificare și urmărire a instrucțiunilor complexe.

44

```
ChatGroq(model="llama-3.3-70b-versatile", temperature=0)
```

Pentru ambele niveluri de procesare, parametrul temperature este setat explicit la valoarea 0. Într-un workflow bazat pe grafuri (StateGraph), consistența structurii datelor (JSON output) și predictibilitatea deciziilor de rutare sunt vitale. Orice variație creativă ("halucinație") poate duce la eşecul tranzitiei între noduri.

Nodul Extractor

Rol: Analizează input-ul utilizatorului (limbaj natural) și extrage entitățile relevante de tip skill, necesare inițierii fluxului RAG.

Flux de Execuție:

- Input-ul este vectorizat folosind modelul de embedding
- Se interoghează Semantic Cache-ul (Redis) utilizând similaritate cosinus
- Prag de similaritate: 0.15
 - permite variații moderate de exprimare
 - menține precizia extractiei
 - ex: „Sunt programator Python” ≈ „Lucrez cu Python de 5 ani”

Gestionarea Cache-ului:

- CACHE_HIT: skill-urile sunt returnate instantaneu
- CACHE_MISS: se construiește un lanț LangChain:
- Rezultatul este stocat în cache împreună cu vectorul interogării
- Interogările similare ulterioare devin CACHE_HIT



Agentul Advisor

Agentul Advisor reprezintă puntea dintre profilul de competențe al utilizatorului și baza de cunoștințe organizațională (Neo4j).

Scopul său este identificarea celui mai potrivit rol profesional folosind un criteriu de tip Best Fit.



Strategie de Caching pe Două Niveluri

Nivel 1 – Semantic Cache (Strict, threshold = 0.1)

- Interogarea este vectorizată și comparată semantic
- Prag foarte strict pentru a evita confuzii semantice
 - ex: Java ≠ JavaScript, React ≠ React Native
- Previene „halucinațiile” cache-ului semantic
- Returnează rezultate doar pentru combinații de skill-uri extrem de similare

Nivel 2 – Exact Cache (Fallback)

- Se verifică o cheie deterministă bazată pe lista de skill-uri sortată
- Protejează împotriva recalculărilor costisitoare pentru input-uri identice
- Acoperă cazuri rare în care potrivirea semantică nu trece pragul strict

În cazul unui CACHE_MISS, sistemul interoghează graful Neo4j folosind o strategie de agregare:

- Nu se caută potrivire perfectă
- Se caută rolul cu cea mai mare suprapunere de skill-uri

Agentul Scout

- **Agentul Scout** reprezintă motorul principal de căutare al sistemului, responsabil de identificarea oportunităților reale din piața muncii.
- Implementează o arhitectură hibridă: Vector Search + Metadata Filtering.

Validarea Parametrilor de Intrare

Înainte de inițierea căutării:

- Se verifică prezența parametrilor critici: location, experience_level
- Dacă lipsesc, funcția returnează un mesaj de tip STOP, nu o eroare tehnică
- LLM-ul este ghidat să solicite informațiile lipsă utilizatorului
- Beneficiu: Previne interogări vagi și rezultate irelevante.



Strategie de Caching Hibrid

Nivel 1 – Exact Cache (Primar)

- Se generează o cheie deterministă din payload-ul normalizat
 - Dacă există (EXACT_HIT), răspunsul este returnat instantaneu
 - Evită complet calculul de embedding și căutarea vectorială
-
- La Semantic Cache HIT, rezultatul este salvat automat și în Exact Cache
 - Următoarea interogare identică devine EXACT_HIT

Interogarea MongoDB (CACHE_MISS)

Vector Search:

- Se utilizează indexul job_vector_index
- Căutare bazată pe similaritate semantică a descrierilor

Post-Filtering determinist:

- Filtrare strictă pe:
 - locație
 - nivel de experiență

Nivel 2 – Semantic Cache (Fallback, threshold = 0.15)

- Permite variații mai mari de exprimare
- Necesare datorită complexității input-ului (titlu + locație + experiență)
- Optimizează performanța pentru interogări similare, dar nu identice

Fallback Logic

- Dacă vector search nu returnează rezultate:
 - se comută automat pe căutare clasică (find)
- Asigură robustețe și disponibilitate a sistemului



Monitorizare și Observabilitate



ARHITECTURA DE MONITORIZARE

Sistemul utilizează un stack standard, production-grade:

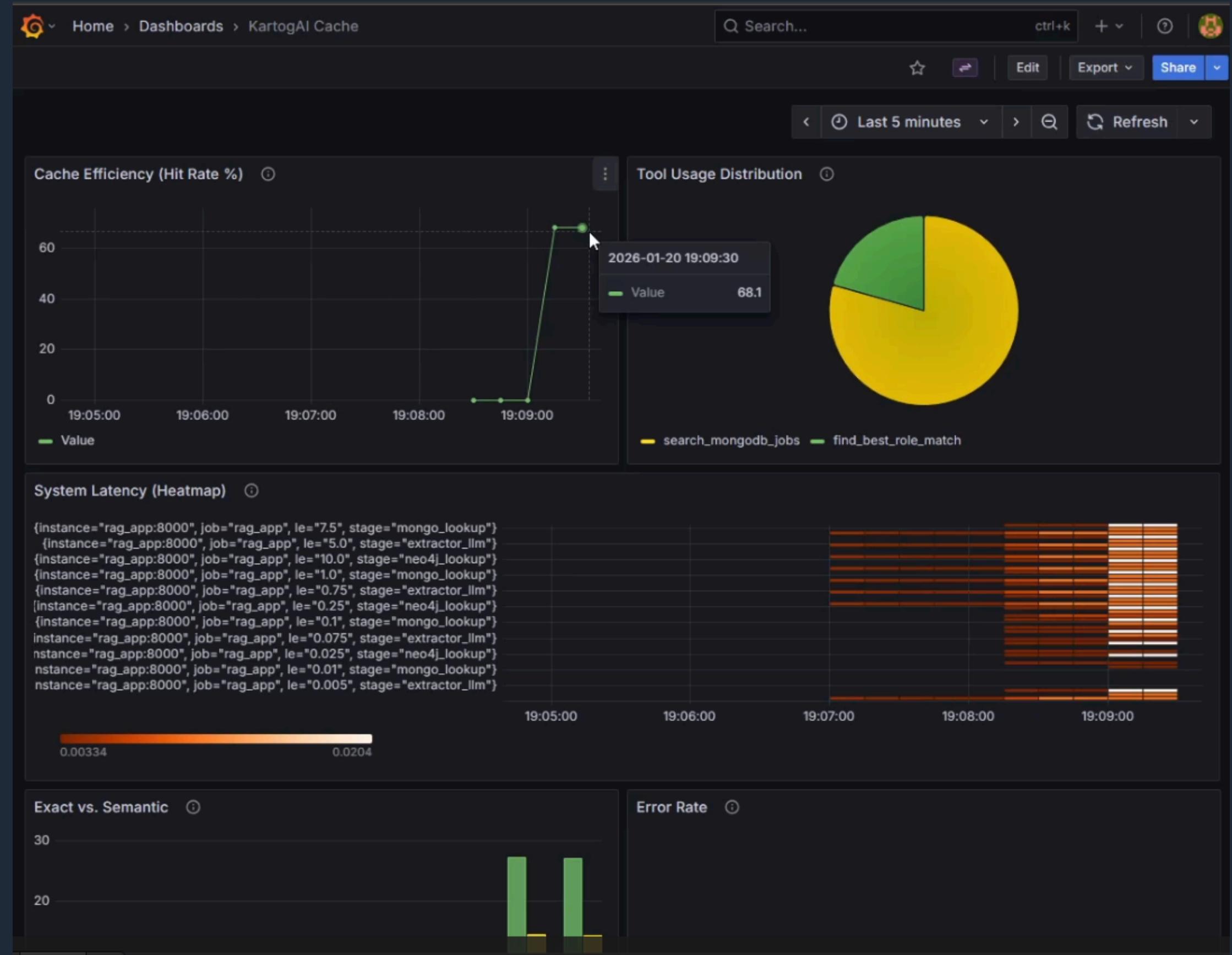
- Prometheus – colectare și stocare metrii
- Grafana – vizualizare și analiză metrii

Aplicația Python expune un endpoint HTTP de metrii pe portul 8000, rulat într-un thread separat (daemon=True), pentru a nu afecta fluxul principal al aplicației Streamlit.

METRICI COLECTATE

- kartog_cache_ops_total (Counter)
Monitorizează eficiența caching-ului
Etichete: - method: exact / semantic
- status: hit / miss
- kartog_tool_usage_total (Counter)
Măsoară frecvența de utilizare a agentilor și tool-urilor (ex: find_best_role_match, search_mongodb_jobs)
- kartog_request_latency_seconds (Histogram)
Măsoară latența etapelor critice:
 - extractor_llm
 - neo4j_lookup
 - mongo_lookupUtil pentru identificarea bottleneck-urilor de performanță

Grafana Dashboard



Vă mulțumim!

