

A Conversational AI Agent for FIB

Hierarchical Design and LLM-as-Judge Evaluation

Ákos Schneider Ignasi Cervero

Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Human Language Engineering

The Problem: Fragmented Academic Information

Students face multiple challenges:

- Data scattered across Racó, FIB API, and website
- Simple queries require many clicks
- No system understands implicit context
- Real-time info critical during exams

Example: “When is my next exam?”

- Check enrolled courses
- Navigate to exam section
- Filter by your courses
- Compare dates manually



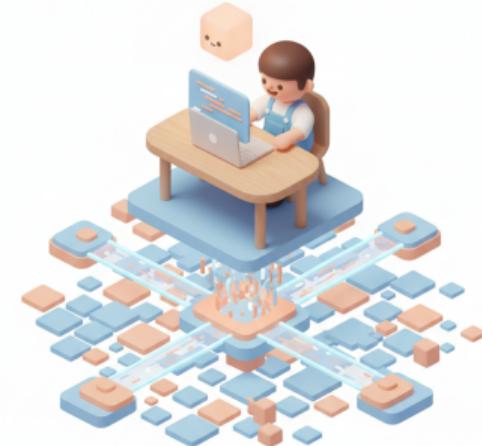
Our Solution: Natural Language Interface

A conversational AI agent that:

- Understands natural language queries
- Integrates with the official FIB API
- Handles implicit context automatically
- Supports both public and private data

Now you can just ask:

- “When is my next exam?”
- “What do I have tomorrow?”
- “How many credits is IA?”
- “Who teaches EDA?”



Architecture: Hierarchical Agent Design

Two-level hierarchy:

- **Root Agent**

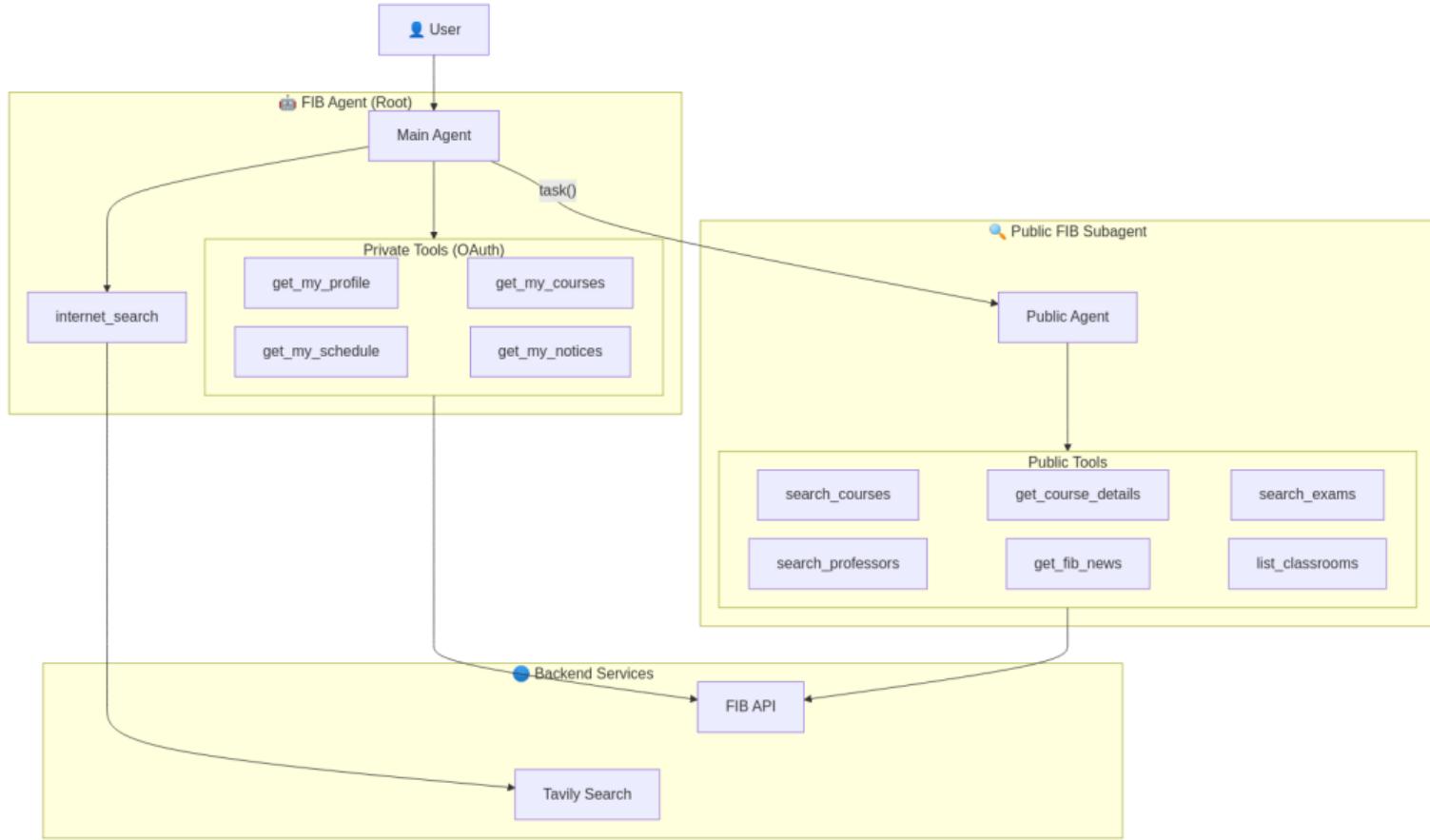
- User context & authentication
- Private tools (OAuth)
- Internet search fallback
- Task delegation

- **Public Subagent**

- Specialized for FIB API
- Focused system prompt
- Public data queries

Benefits: Focused prompts, security boundary, modularity





The Tool Ecosystem

Public Tools (FIB API)

- `search_courses` – Find courses by name/code
- `get_course_details` – Full course info
- `search_exams` – Exam schedules
- `search_professors` – Faculty search
- `get_fib_news` – Announcements
- `list_classrooms` – Room info

All tools are typed Python functions with Pydantic validation

Private Tools (OAuth required)

- `get_my_profile` – User info
- `get_my_courses` – Enrolled courses
- `get_my_schedule` – Personal timetable
- `get_my_notices` – Course notices

External

- `internet_search` – Tavily API

How It Works: The ReAct Loop

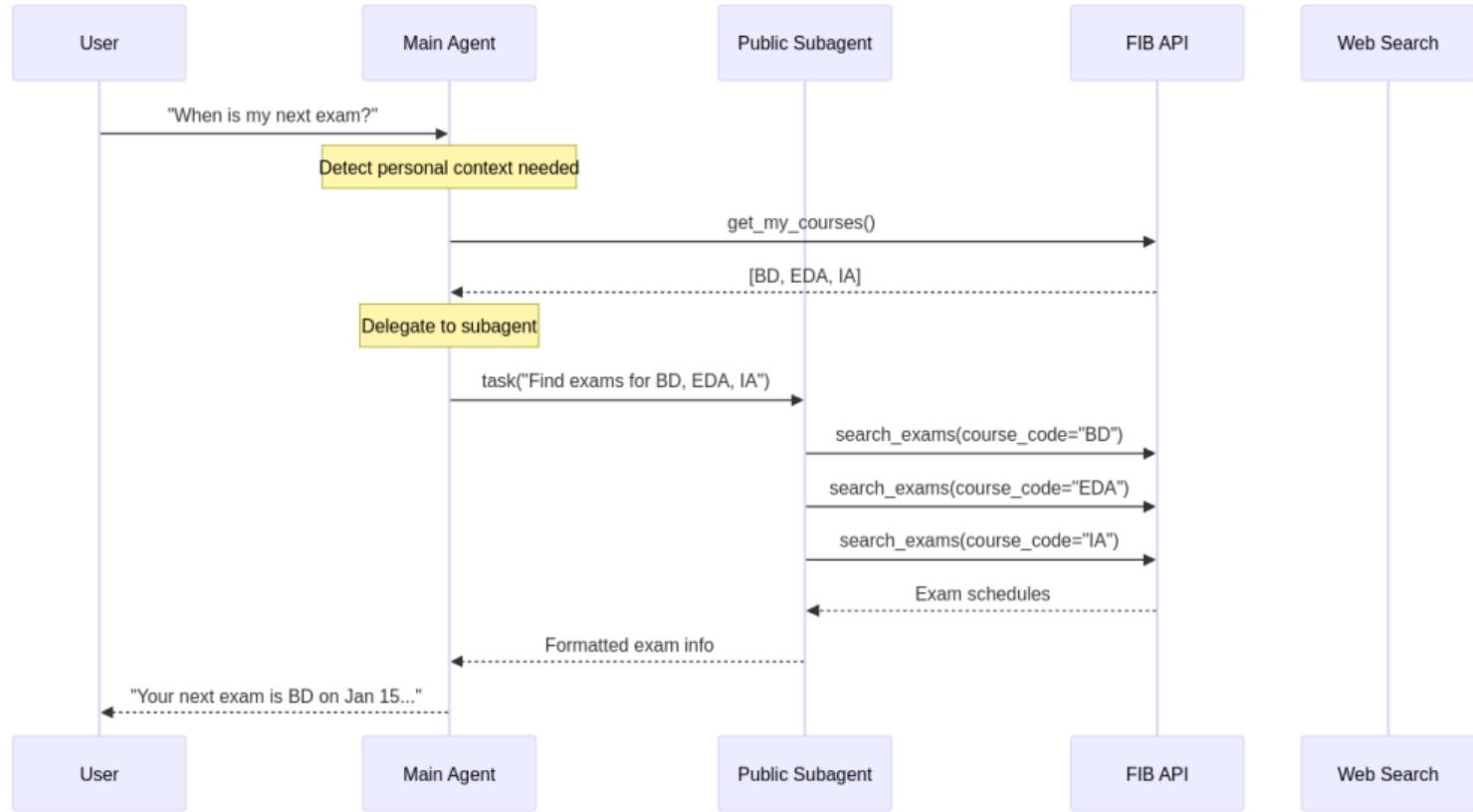
Reasoning + Acting paradigm:

- ① **Thought** – Reason about the query
- ② **Action** – Call appropriate tool
- ③ **Observation** – Process result
- ④ **Repeat** until answer ready

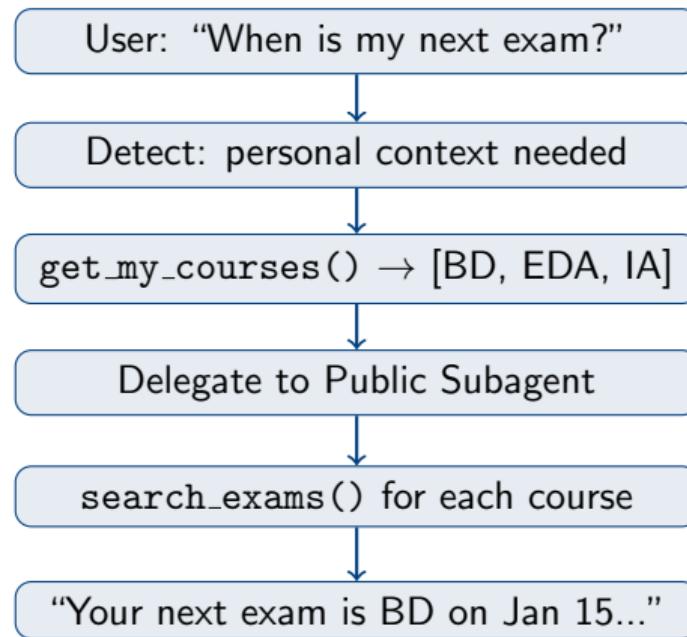
Example thought process:

- “User wants exam info”
- “Need their enrolled courses first”
- Call `get_my_courses()`
- “Got [BD, EDA, IA], now search exams”





Example: “When is my next exam?”



Prompt Engineering Highlights

Key techniques in system prompts:

- **Context detection**
 - “my exam” → check enrolled courses first
 - “tomorrow” → resolve to weekday
- **Date reasoning**
 - Explicit weekday mappings (Monday=1)
 - Current date in context
- **Disambiguation strategy**
 - “Machine Learning” → multiple matches
 - Present top options, ask for clarification
- **Quality checklist** before responding



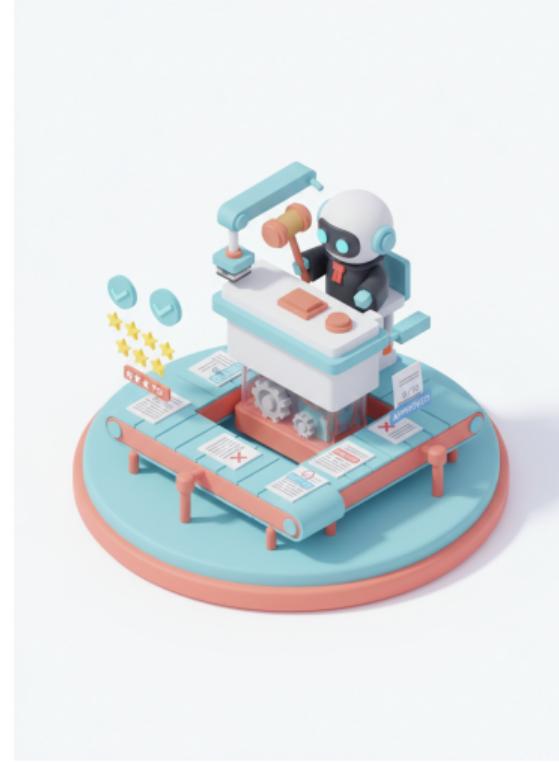
Evaluation: LLM-as-Judge Framework

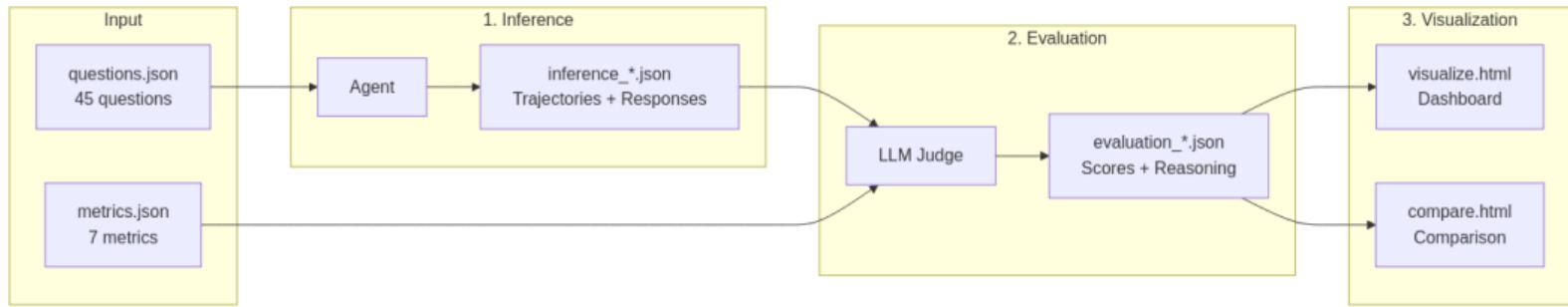
Why LLM-as-Judge?

- Traditional metrics (BLEU, ROUGE) correlate poorly with quality
- LLM judges understand semantic meaning
- Custom rubrics for each metric

Our setup:

- 45 curated test questions
- 10 categories of queries
- 7 evaluation metrics
- Gemini 2.5 Flash Lite as judge





Question Categories

Category	Count
courses	13
exams	5
professors	4
personal (OAuth)	4
multi_tool	5
ambiguous	4
news, academic, etc.	10

Complexity: Simple, Multi-step, Contextual, Ambiguous

7 Evaluation Metrics

- **Relevance** – addresses the question?
- **Helpfulness** – actionable & useful?
- **Conciseness** – appropriately brief?
- **Structure** – well-organized?
- **Tone** – professional?
- **Error Handling** – graceful failures?
- **Tool Appropriateness** – right tools?

Results: Meeting Our Targets

Gemini 2.5 Flash Results

Metric	Score	Target	
Relevance	0.90	>0.85	✓
Helpfulness	0.89	>0.85	✓
Conciseness	0.85	>0.80	✓
Structure	0.77	>0.75	✓
Tone	0.85	>0.80	✓
Error Handling	0.51	>0.70	✗
Tool Approp.	0.77	>0.80	~~
Average	0.79		

Key Findings

- Core metrics (relevance, helpfulness) exceed targets
- Flash vs Pro: nearly identical scores
- Flash recommended: faster, cheaper

Top Categories (Relevance)

- courses: 0.95
- academic: 0.95
- ambiguous: 0.93
- exams: 0.92

Lessons Learned

Challenges & Solutions

- **Ambiguous queries**

- Present top matches first
- Then ask for clarification

- **Date-relative queries**

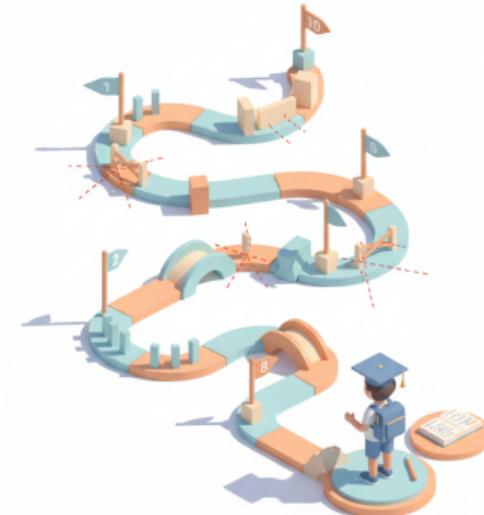
- Explicit date in context
- Weekday mappings in prompt

- **Error handling scored low**

- Rubric was overly strict
- Agent too verbose on errors

- **Tool tracking across subagents**

- Evaluation sees only “task” call
- Need trajectory flattening



Contributions & Future Work

Contributions

- Open-source FIB Agent with hierarchical architecture
- 45-question evaluation dataset (10 categories)
- Reusable LLM-as-judge framework
- MCP server for AI interoperability
- Documented prompt engineering patterns

Future Directions

- RAG with syllabi & lecture notes
- Conversation memory (multi-turn)
- User interface (Telegram bot, web chat)
- Multi-university adaptation



Thank You

Key Takeaway

“Modern LLM agents can effectively serve as natural language interfaces to structured APIs when properly configured with domain-specific tools and engineered prompts.”

Links

- FIB API: <https://api.fib.upc.edu/v2/>
- LangGraph: <https://github.com/langchain-ai/langgraph/>
- deepagents: <https://github.com/langchain-ai/deepagents>

