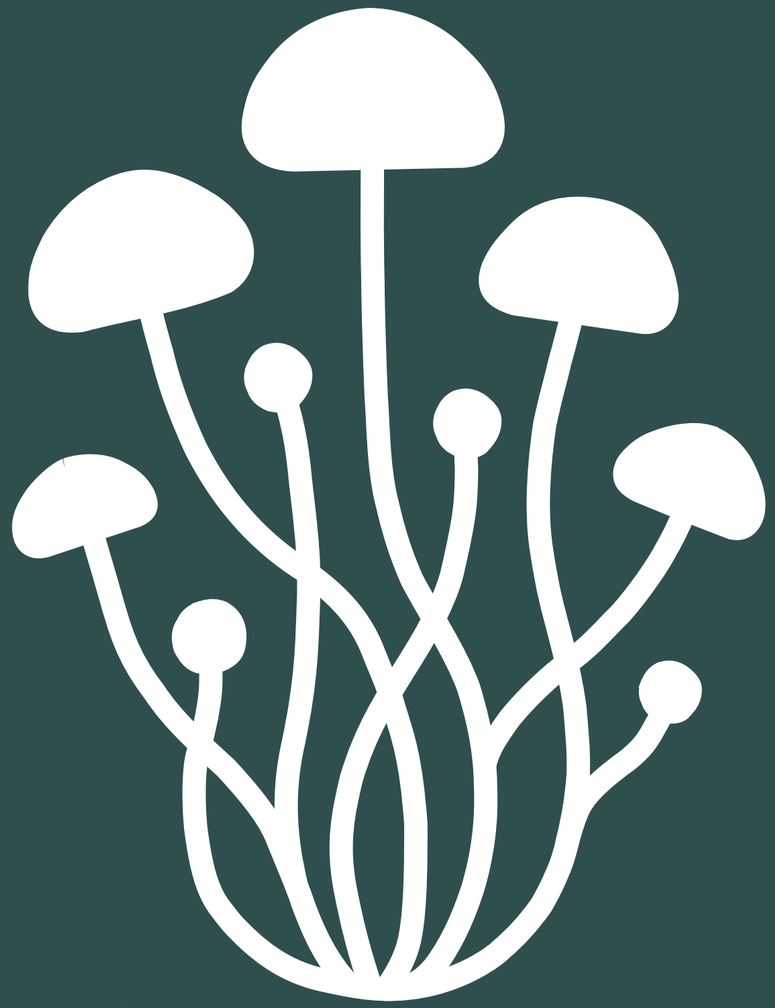


Gaia: building Geo-AI on an open stack

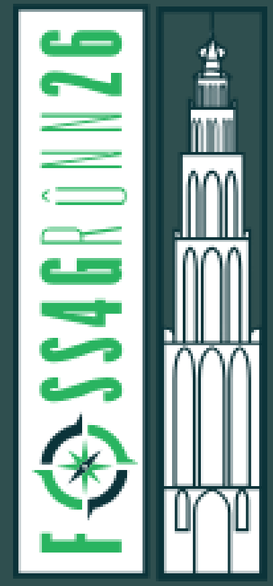
FOSS4GNL — Groningen

Kamiel Verhelst & Sytze Simonse

9th of July, 2026

enoki

AI, rooted in values



What’s already there?

GeoAI

Geospatial prompting

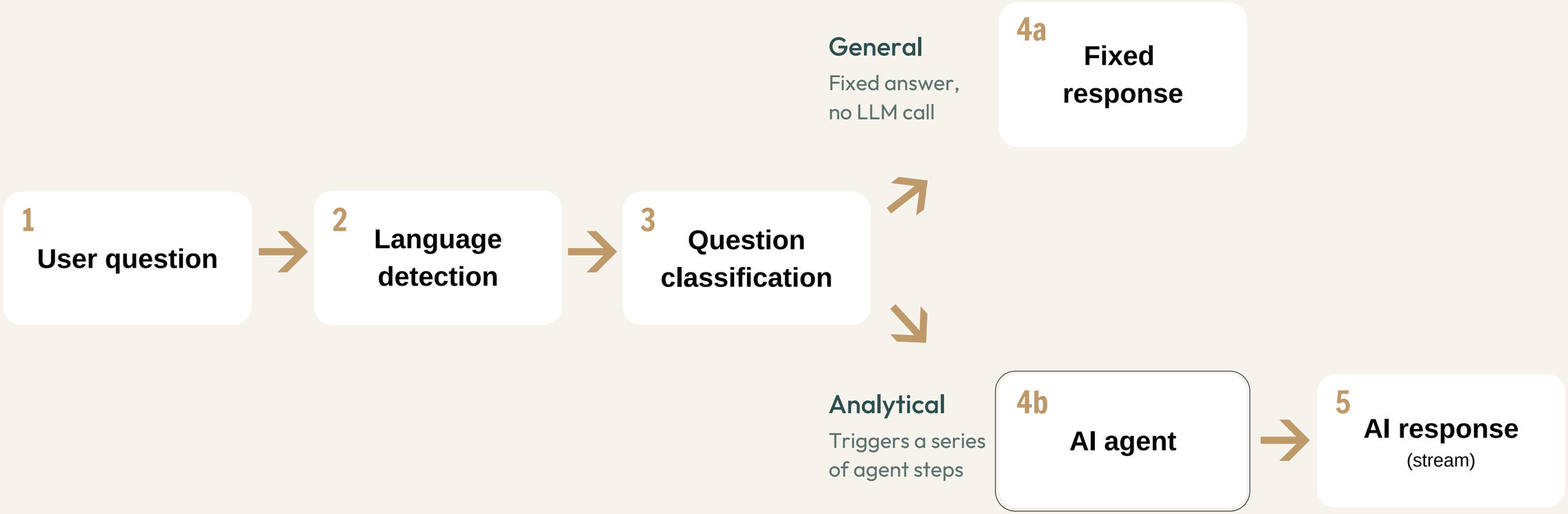
Agentic AI

Demo

gaia.enoki-ai.nl

BUILDING AN AI AGENT

From question to answer



BUILDING AN AI AGENT

Steps of the AI agent

1. Fetch metadata for the layers in the session
2. Fetch the schema of the relevant tables AI
3. Build and execute a SQL query on PostGIS AI
4. Execute query. On fail: retry 3 (max. 3 attempts)
5. Form answer in text, including rendered UI elements AI

BUILDING AN AI AGENT

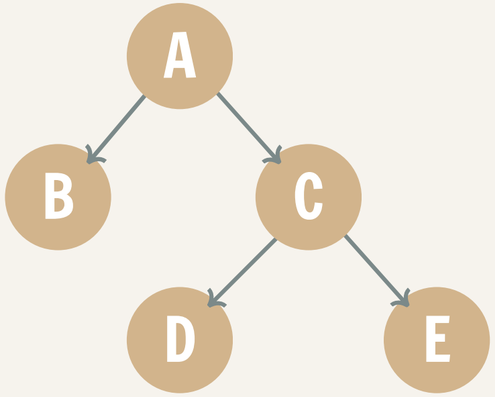
How to build agent logic?

LangChain

The framework: building blocks for prompts, tools and memory around an LLM call.

LangGraph

The control flow: an explicit graph of steps the agent moves through, calling our tools itself. Conditionals control path.



INFRASTRUCTURE

Data storage & catalogue

Every layer arrives as vector — points lines or polygons — and lands, regardless of source format, in the same PostgreSQL + PostGIS database.

1 Storage: PostgreSQL + PostGIS

2 Conversion: GDAL (ogr2ogr) as spatial engine

3 Display: visualised as GeoJSON

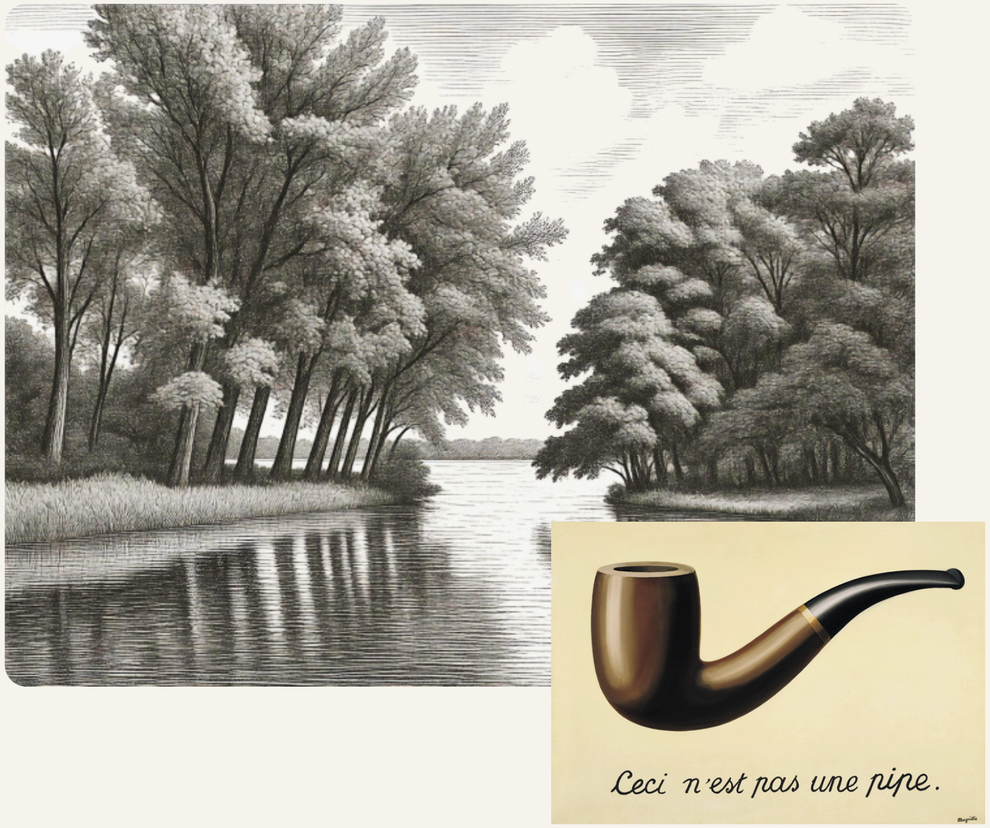
Per layer, also stored in the database:

- filesize
- filename
- upload timestamp
- column names
- geometry type
- extent

AI & GEO

Semantic correctness

Retries only catch crashes — a wrong column name, a syntax error. A query that runs, but is **semantically** wrong is not caught. Does the model understand what a “park” actually is?




```
...
### RULE 2: PARK IDENTIFICATION (EXACT FILTER)
When asked about "parks":
  - Filter: LOWER(name) LIKE '%park%' AND LOWER(name)
NOT LIKE '%polder%'
  - EXCLUDE: forest, polder, strip, nature (unless
explicitly asked)
  - A "park" must have "park" in its name
...
```

Prompt engineer reliable?

```
### ABSOLUTE RULES — NEVER VIOLATE
1. Always respond in {lang_upper}
2. If the database errors, fix the query
3. Never invent data — say "I could not find the
answer"
4. Never fabricate numbers, names or statistics
5. If a query fails twice, admit the limitation

### ALWAYS QUERY DATA FIRST
- Never say "I can't help" without querying first
- First call get_session_datasets_metadata
- Search columns with LIKE '%searchterm%'
- Only say "not found" after 0 results

### COMMUNICATION RULES
- No technical jargon (SQL, ST_Area, JOIN…)
- No code shown to the user
- Human language only, like a local expert
- Round numbers, 1 decimal place
```

PROMPT ENGINEERING & MODEL CHOICE

Switching between models

Via the environment variable LLM_PROVIDER — a deploy-time choice, not something that changes per question.

Claude

Paid, reliable — but can be expensive and date leaves to third-party servers..

PRACTICAL DEFAULT

OpenRouter

Broad choice of models, easy to switch — but data leaves to third-party servers.

ALTERNATIVE

Ollama

Local, complete data sovereignty and control — but self-hosting models requires good hardware.

SOVEREIGN DATA

When is a model open?

LOOKING AHEAD

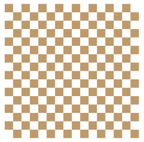
What's next

Improve guardrails

Further integration with UI

Experiment with open source models

Thorough testing with users



Integrate raster support

Any ideas?

THANK YOU

Questions?

Check out ‘Gaia’ on GitLab
gitlab.com/enoki-ai/gaia

Follow us on LinkedIn
linkedin.com/company/enoki-ai-nl

What would you (not) use GenAI for in your geo applications?

What would be the best use of GenAI in your field of work?

What problems do you foresee in using GenAI with geodata?

How would you finetune an LLM for geospatial tasks?