The AI Stack
We Actually Run.

Nick Finch
Chief Technology Officer, inmydata

PUG Challenge 2026, Amsterdam



Every developer in this room has a frontier model. Almost nobody has lived with one in production.



inmydata



Studio: Self-Service Analytics

inmydata Studio. An agentic dashboard designer.

The user talks their way to a dashboard.

inmydata

840 manager days a year,
spent preparing for a conversation.

Fraser Hart and Fields. 35 jewellery stores.

Every period, every store manager reviews every colleague, and each spends two days pulling the figures from our dashboards first.



Buying green coffee

Segafredo Zanetti Nederland. A roaster buying a volatile commodity months ahead.

Supply depends weather, crop yields, transport costs. Demand sits in their own ERP.

The people who do it well hold all of that in their heads.

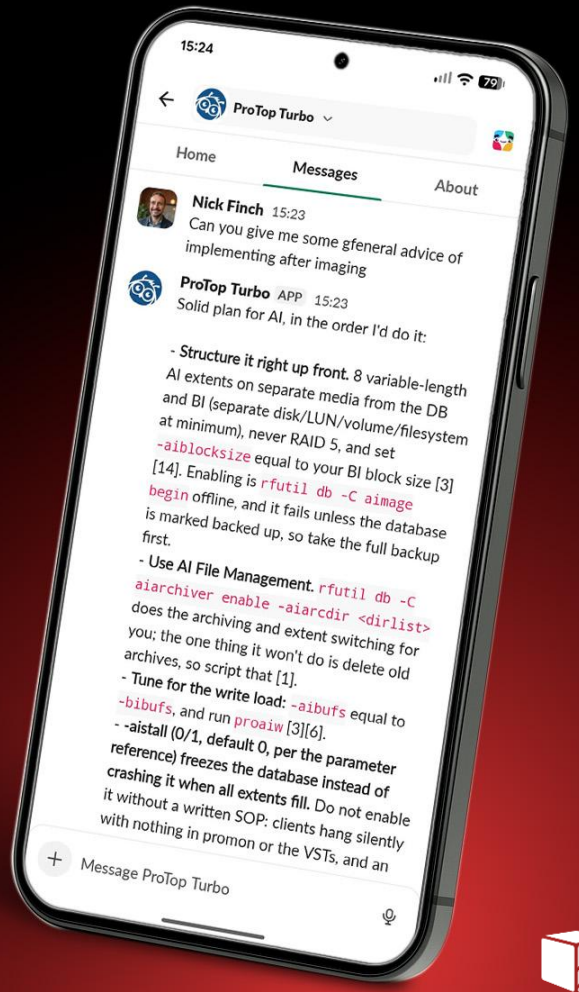


From a window to a caretaker

Protop a monitoring product with dashboards, alerts, diagnostics. A window on the database for an experienced DBA.

Stage one. The agent offers expert advice inside the product.

Stage two. A trusted path for the agent to act, and fix the issue itself.



inmydata

This is the bill of materials.
Four kinds of data, four stores, one door, one mouth.

AI-First in Practice was the argument. This is what we actually run: where each kind of data lives, which models and tools sit on it, what we tried and abandoned, what it costs, and where the model still can’t be trusted. The examples come from outside OpenEdge. The guidance is for plugging each one into it.

inmydata

Four kinds of data. One mouth.

Structured

Commands, parameters, alerts, metrics. An RDBMS and a fixed set of tools. The model picks a tool and reads the rows.

Analytics

Large volumes, aggregated fast. An analytics platform, a semantic layer, a typed query. The model asks, the engine answers.

Knowledge

Documents, interviews, transcripts. A vector store. The model answers from text it has read.

Governance

Rules in Markdown, pushed into the system prompt. The model answers within them.

What the model answers from

What it answers within

inmydata

Structured data

ProTop. The OpenEdge commands and their parameters. The alerts and metrics ProTop raises, and what each one means. Tables, not prose.

The agent asks a precise question and gets rows back. What does this parameter do? Which alert fired, and what is its threshold?

The simplest kind of data, and the one most OpenEdge applications already have.



The stack we run: structured data

The store

Postgres. An RDBMS is exactly right for this: small, precise, normalised, looked up by key. OpenEdge over an AppServer call works the same way.

The tools

A fixed set of dedicated API endpoints, each one a tool the agent can call. No MCP server in front of them.

The models

Haiku for classification, deciding what kind of question this is. Sonnet for the rest. Anthropic only, other models configurable in the front end.

inmydata

The assumption we dropped

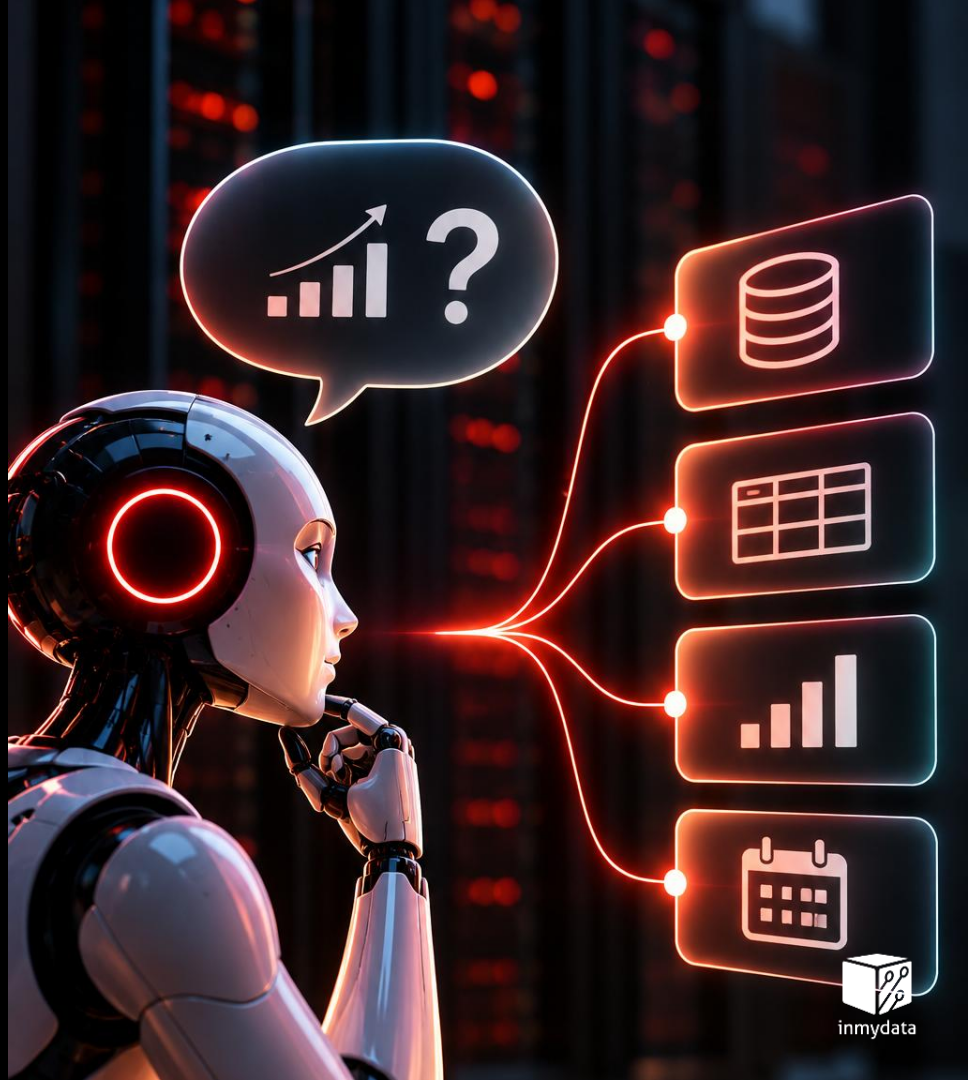
Everything behind an MCP server

MCP is the natural assumption, and for the analytics platform it is the right one. But MCP costs tokens: on every turn the model has to reason over the tools on offer.

These APIs are static. A fixed set of tools, described once, works better and costs less. Changing them is not hard.

The door should match the data. A general door for a general store. A fixed set of tools for a fixed set of tables.

The ProTop expert: Postgres, dedicated endpoints, no MCP.



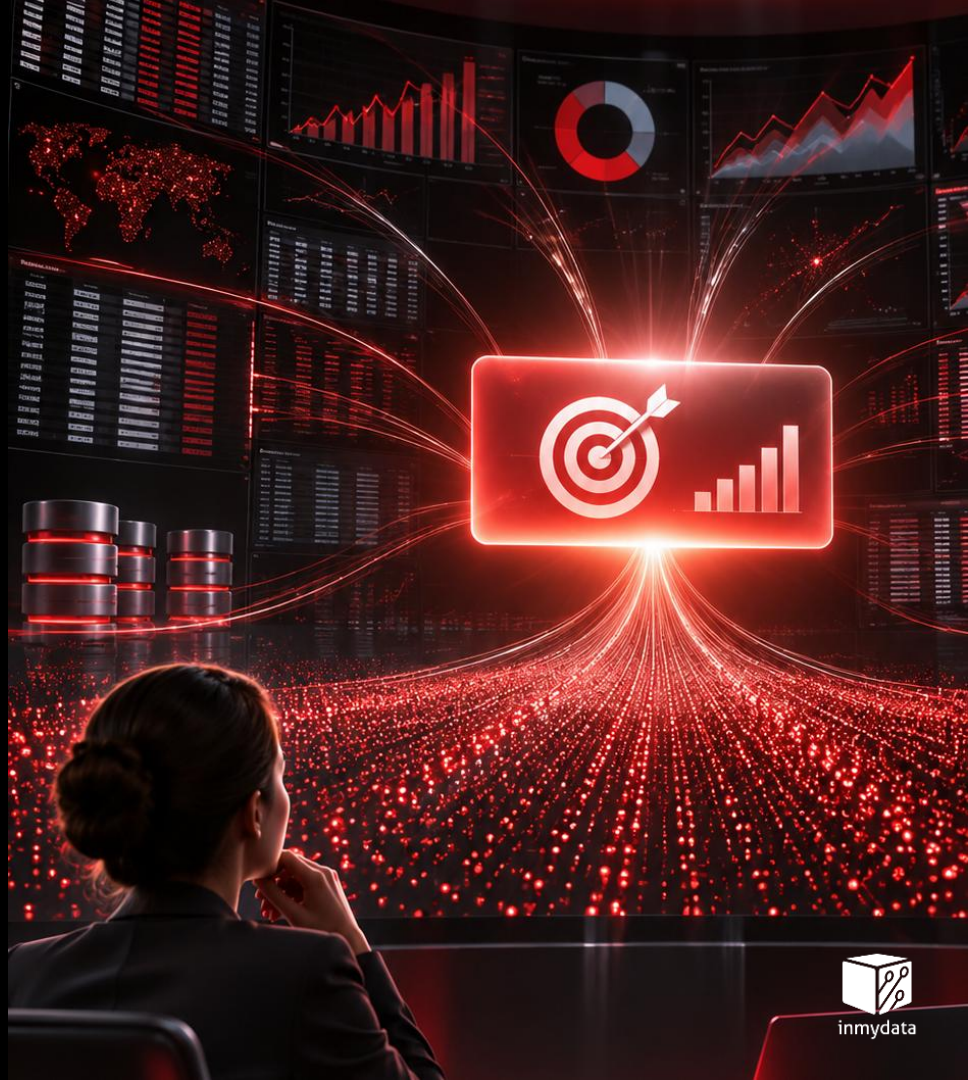
The model picks a tool and reads the rows.
That is the whole trust model.

Where it can’t be trusted: with the parameters. Validate every tool call at the boundary, return the error to the agent, and let it try again. For OpenEdge: expose the lookups you already have as a handful of AppServer-backed tools. No platform, no vector store, no MCP server needed to start.

Analytics data.
Millions of rows, one question.

Studio, the ANG agent and the TikTak buyer all live here: large volumes of operational data, pulled and aggregated fast enough to hold a conversation.

Haiku and Sonnet do the asking in Studio. Opus where the answers feed a decision.



The stack we run: analytics data

The feed

A scheduled .NET and ABL task over the AppServer today. Web APIs on the new platform: push, SFTP drop or scheduled pull. 26 datasets for TikTak.

→

The store

StarRocks per tenant, Postgres catalogue. Millions of rows in a fraction of a second. Big aggregates land in a temporary DuckDB the model can query.

→

Semantic layer

Subjects, metrics, 4-4-5 calendars, relationships, calculations in a constrained SQL subset. Twenty years of business logic.

→

The door

A FastMCP server generated from the API contract. Nine tools. Token exchange so the agent acts as the user: row security, masks, audit on every read.

inmydata

Tried and abandoned

Querying the live database

DataPA's original design pulled report data straight from the Progress database over the AppServer. Real-time analytics, no copy.

It worked for years, then it didn't. A row-based, normalised store is built for transactions, and retrieval over large volumes is slow by design. We moved to a hosted warehouse and never went back.

The lesson transfers unchanged. Conversations don't survive a thirty-second query.

inmydata has been building analytics and data platforms since 2003.



Plugging it into OpenEdge.
The AppServer is the feed, not the query path.

Push or pull the data out of OpenEdge into a columnar store that answers in milliseconds. Put a semantic layer over it. Give the agent a typed query interface through MCP, never a database connection. Keep the AppServer for the transactional work it is good at.



Knowledge data

The DBA expert system for ProTop. Decades of tacit knowledge, documentation and interview transcripts, turned into something a model can answer from.

ANG and TikTak carry the same kind: the judgement of a People Director or a coffee buyer, captured by interview.

The model is asked to judge and reason here. The highest hallucination risk, and the most engineering around retrieval and provenance.



The stack we run: knowledge data

Retrieval

Postgres with pgvector and tsvector, fused with reciprocal rank fusion. Voyage embeddings. Chunk-level citations the model is told to emit.

→

Ingest

Parse, chunk, assess quality, extract entities, embed. Haiku extracts entities from every chunk; Opus consolidates them weekly.

→

Living KB

Expert corrections become auditable semantic patches. Gaps queue from retrieval signals and route to the right expert for an interview.

→

Controls

Row-level security on 27 tables. Six LLM feature switches and weekly budgets per tenant. Sonnet answers; Opus judges; Haiku classifies.

inmydata

“Delete the .lk file and reboot the server.”

Transcripts are the worst input for RAG: noise, side conversations, half-sentences. A model rewrites them into chunks, and the chunks read beautifully. That is the trap.

The model isn’t wrong exactly. It adds the wrong emphasis or loses the condition. The knowledge was real. The “when” had been parsed away.



Provenance with a trust state

Verified by the expert

Interview chunks are checked by the interviewee before they load. The human in the loop is the person whose knowledge it is.

Unverified

Everything else enters flagged. The agent is told never to use unverified knowledge in a customer-facing answer, only inside White Star's own Slack, where experts are primed to push back.

Earned

Once a chunk has appeared in front of an expert enough times and been waved through, the flag comes off. Knowledge earns its way to the customer.

inmydata

Governance data

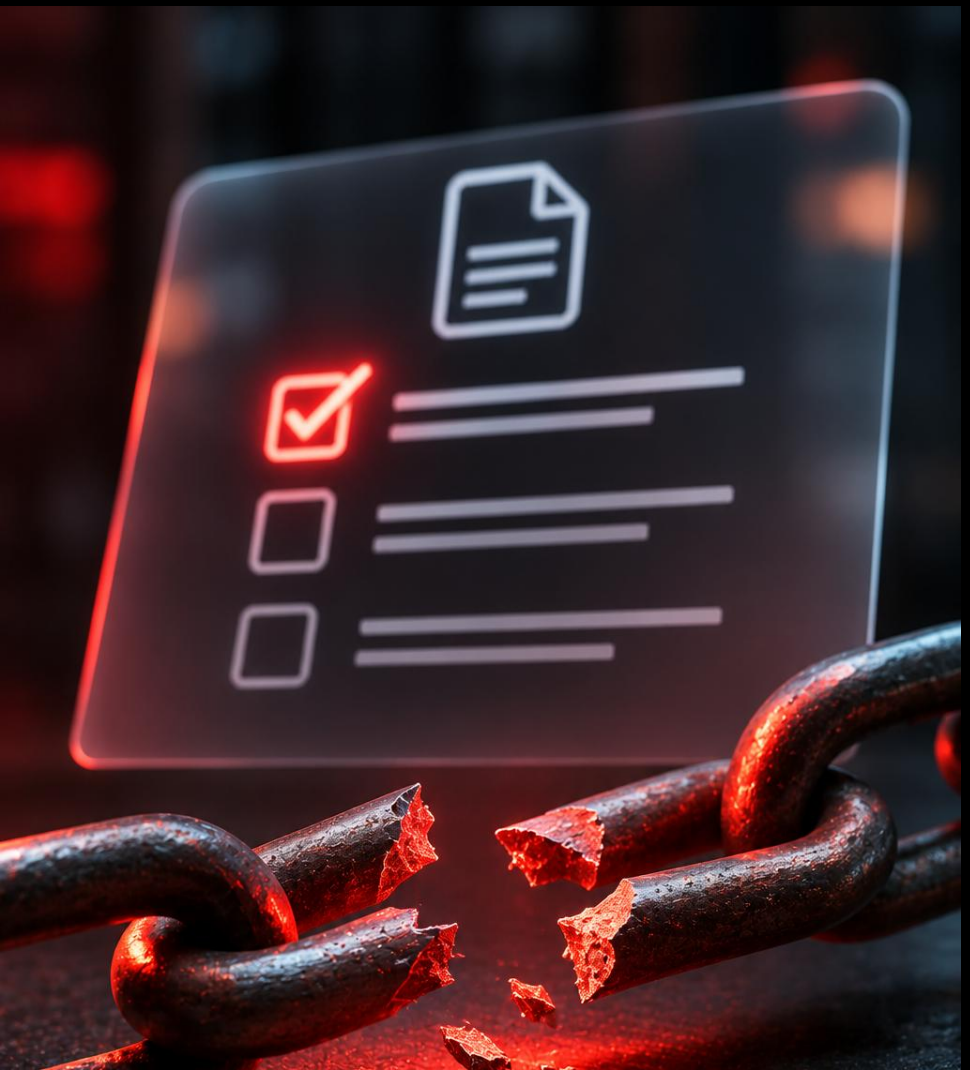
ProTop: if the knowledge base returns nothing, answer “I don’t know”. Never use unverified knowledge with a customer.

ANG: facts only in the brief. No subjective judgement. Never quote the SOP from memory. Wellbeing notes are not input.

TikTak: the buyer’s hard rules. The lines the recommendation must not cross.

Not what the model answers from. What it answers within.





An instruction is not enforcement

The model was told to check OpenEdge commands, to always verify them when cited against the structured data.

That was not enough. The solution was to add a deterministic layer, to check the commands against the database tables.

Governance data tells the model what to do. The boundary checks whether it did.

inmydata

The mouth: chat and voice in front of customers

Studio's chat and TikTak's chat in the browser.

The voice interview platform, where the customer is an expert giving up what they know: senior DBAs at ProTop, the HR director at ANG. Try it at our stand.

The chat inside ProTop 5. See it at the ProTop stand, or in Paul's What's New in ProTop 5, Friday 10:15, Athens.

The Slack integration inside White Star. Deliberately not customer-facing yet.



The stack we run: chat and voice

Text chat

LangGraph with SSE streaming for Studio. The same doors as everything else behind it: tools, MCP, retrieval, the system prompt.

→

Voice

LiveKit Cloud and the Agents SDK. Sonnet in the middle. Deepgram Nova 3 in, Aura 2 out, Silero for voice activity. State saved every five seconds.

→

Afterwards

Opus turns the transcript into a topic-sectioned narrative the expert reviews, edits and approves before it goes near a knowledge base.

→

Governance

Consent with timestamp and IP. Audio kept 30 days. GDPR hard delete across S3, database and Cognito. Usage per model and per minute.

inmydata

Tried and abandoned

The smartest model for voice

We tried Opus, Fable, and Sonnet with high thinking in the voice loop. All were too slow, and the conversation lost its natural feel.

Sonnet at medium thinking is the ceiling for speech.

The model choice isn't about capability. It's about the pause before the answer. And voice is billed by the minute, so speed is the cost driver as well as the quality driver.

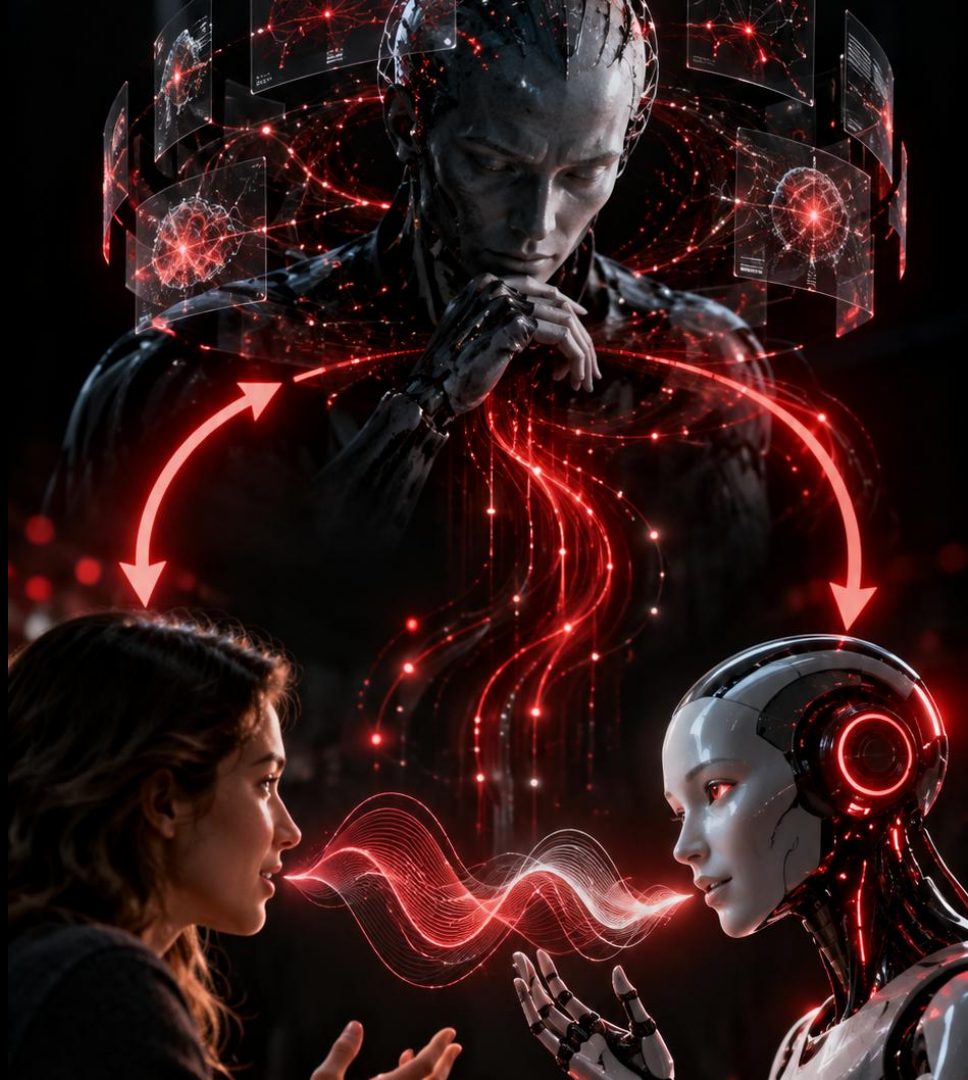
Deepgram failover to AWS speech services: two environment variables and a redeploy.



Latency picks the model.
A second model picks the direction.

For the spec interview platform we want a frontier model driving the conversation: that is what spots the holes and contradictions in a spec.

So Sonnet speaks. Opus or Fable listens to the whole exchange and rewrites the speaking model's context to steer it. The fast model talks. The slow model thinks.





Give the speech engine domain context

Nova 3 hears “AI” and writes “AI”. A DBA means After Imaging. So each interview carries a glossary of up to 200 terms, sent to Deepgram as key terms and to the model as definitions.

inmydata

Costs: Use agent caching – schemas are large

68%

fewer tokens per dashboard build with prompt caching. Roughly 194K down to 62K.

$0.94

per million raw tokens on Sonnet, with an 87% cache-read share. At or below the Haiku baseline.

1 hour

cache TTL. Real users pause longer than five minutes between turns, and the default TTL forced a 38K-token rewrite every turn.

inmydata

A model upgrade needs care

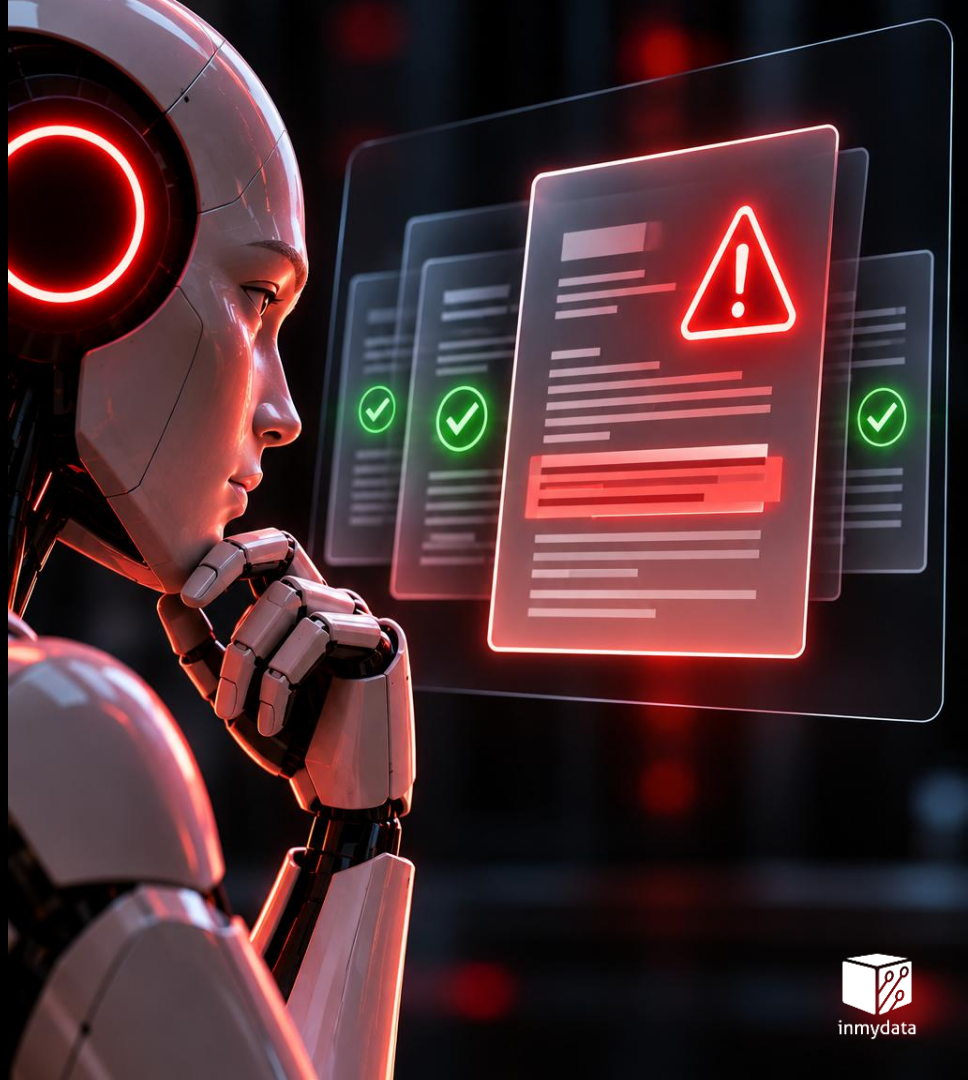
Two production incidents, one structural answer.

A thinking model spent its whole output budget thinking and returned a tool call with empty arguments. Fix: a 32K output budget.

A thirteen-turn exploratory session died on LangGraph’s default 25-step limit.

Always test a model upgrade, don’t assume it will work.

Studio production, August 2026. Customer anonymised.



The stack, for an OpenEdge application

Structured

The lookups you already have, as a handful of AppServer-backed tools. Validate the parameters. Start here.

→

Analytics

Feed OpenEdge into a columnar store and a semantic layer. One typed door, through MCP, as the user.

→

Knowledge

Interview the people. Parse the documents. Verified, unverified, earned.

→

Governance

Rules in Markdown, versioned with the application. Boundaries enforced at the tool edge.

What the model answers from

What it answers within

inmydata

Everyone has the model.
The stack is the part you build.

Four kinds of data. Start with the structured data you already have and a handful of tools. Everything after that is trust, and trust is built one production incident at a time.

inmydata

Thank you

Nick Finch
nfinch@inmydata.com
inmydata.ai
Try the voice interview
at our stand.

