N8N WORKFLOW DOSSIER

Workflows I designed and built

Mathieu Tellene · 2026 · 22 workflows, 530 nodes · 20 of them in production

Built in Cabify's Operational Excellence team in Spain. Each diagram is drawn from the workflow's real n8n export: node positions, types and connections are unchanged. Node labels are translated to English; client, supplier and people names, internal hosts, credentials and parameters are removed.

Case studies with context and results: mathieutellene.github.io

How to read this

Each diagram is one n8n workflow. Flow goes left to right. Solid lines are the normal data flow; dashed lines connect an AI agent to the model, memory and tools it can use. Where a workflow is very wide it is folded into bands, and numbered markers show where a line continues.

Diagrams are vectors: zoom in on the large ones. Small helper workflows are left out, so some sections show only the main ones.

What this does not show: prompts, credentials, parameters and data. The point is the structure: where the model sits, what it is allowed to touch, and what is plain code around it.

Sections 1 to 4 run in production. Section 5 is the retired first version of COPs Control, kept because the contrast with its replacement is the interesting part. Section 6 is a prototype.

Trigger · AI / LLM · Logic and flow · Data and storage · Google Workspace · Messaging · HTTP and APIs · Code

Contents

SECTION

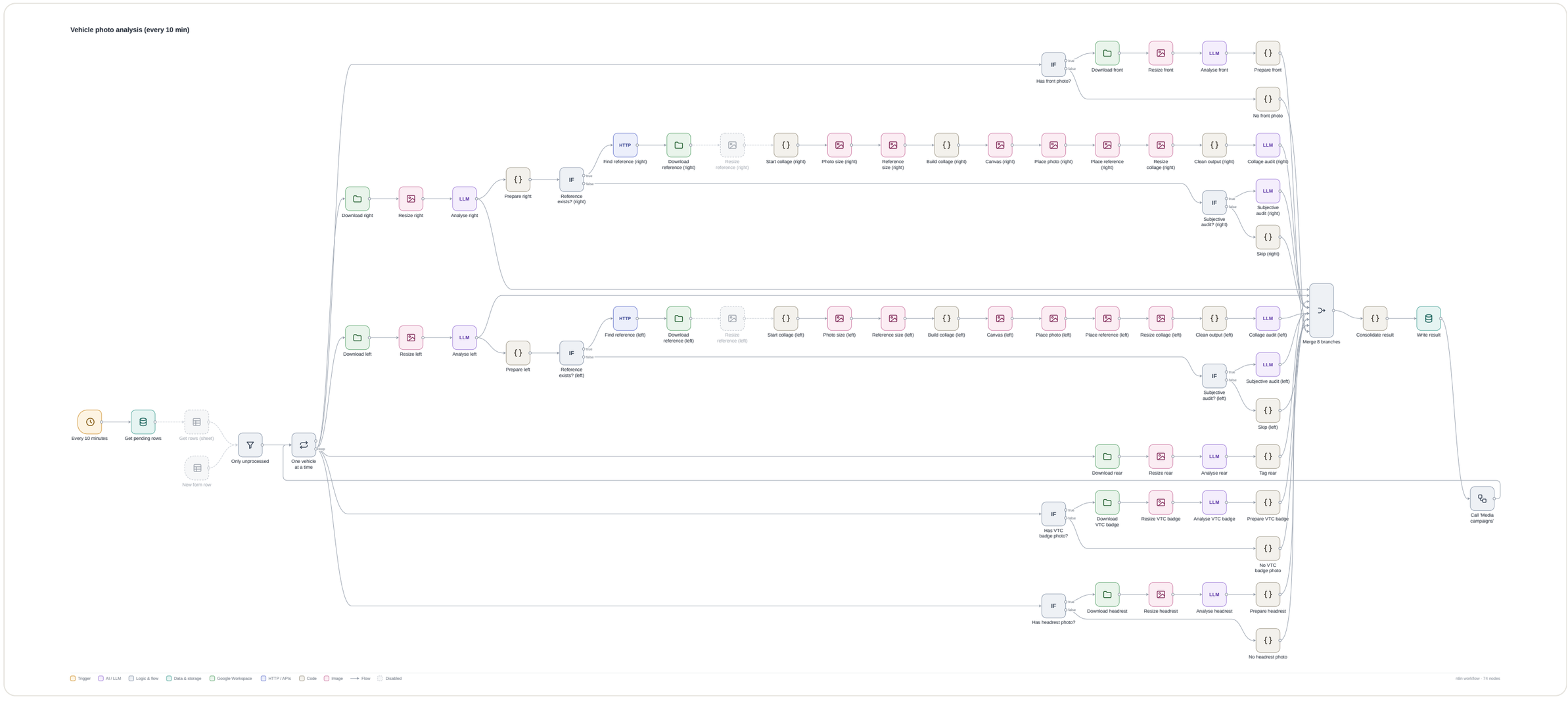
AQM · Asset Quality Management

Drivers and wrapping suppliers photograph each car through web forms. A vision model compares the photos with the reference designs, and the result per vehicle lands in one fleet database that feeds a dashboard and the monthly bonus. 23,621 photos audited at 97% accuracy against human review. Business as usual since 3 September 2026.

Full case study: mathieutellene.github.io/work/aqm.html

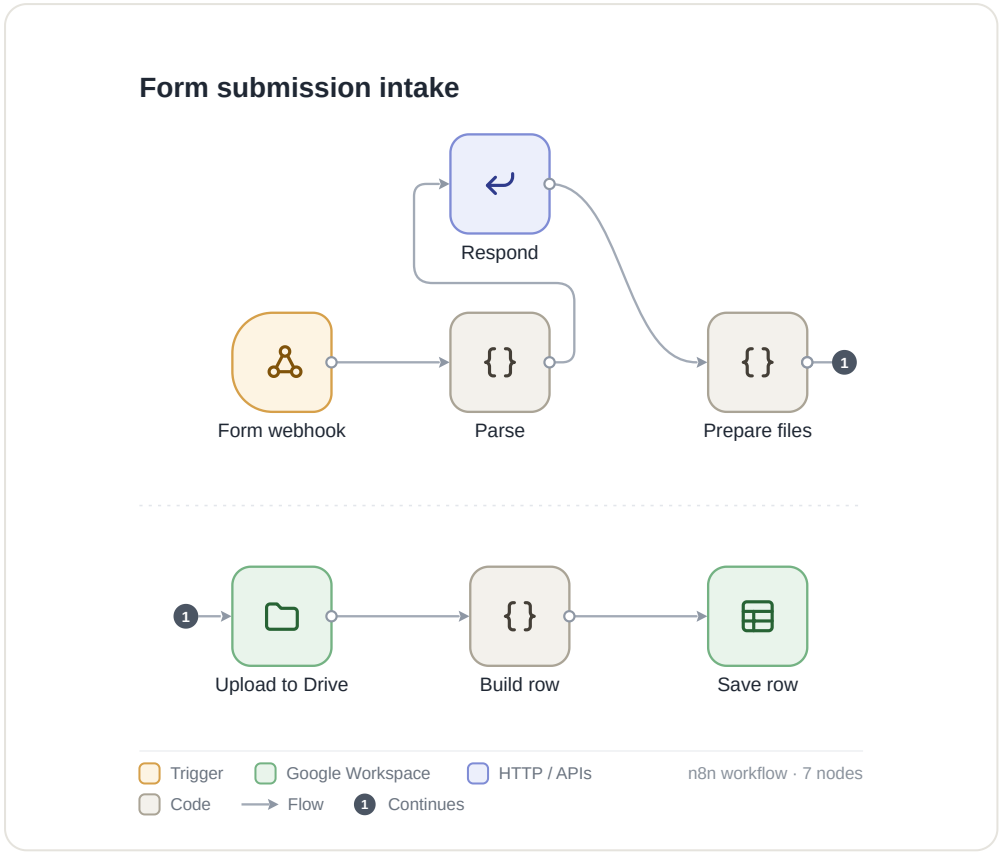
Vehicle photo analysis (every 10 min) `74 nodes`

Runs every 10 minutes and takes one pending vehicle at a time. Each photo (right, left, front, rear, headrest, regional badge) has its own branch: download, resize to 1024 px, ask the model for a fixed 9-field JSON. Side photos also build a collage of the real photo next to the reference design for that vinyl type and side, and a strict comparison audit runs on it; designs without a reference get a subjective audit instead. Eight branches merge into one record, with tokens and cost summed.



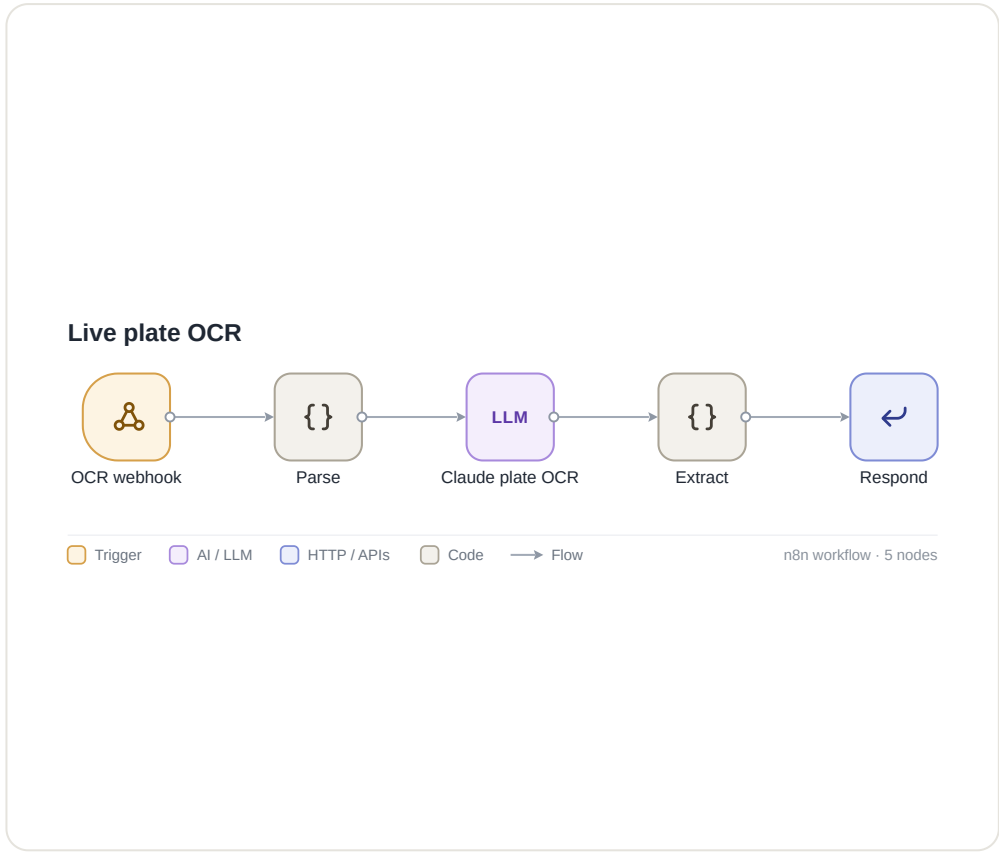
Form submission intake `7 nodes`

The form posts here. Photos are uploaded to one Drive folder per photo type and the submission is written to a data table immediately, so the user gets an answer in seconds while the analysis runs later.



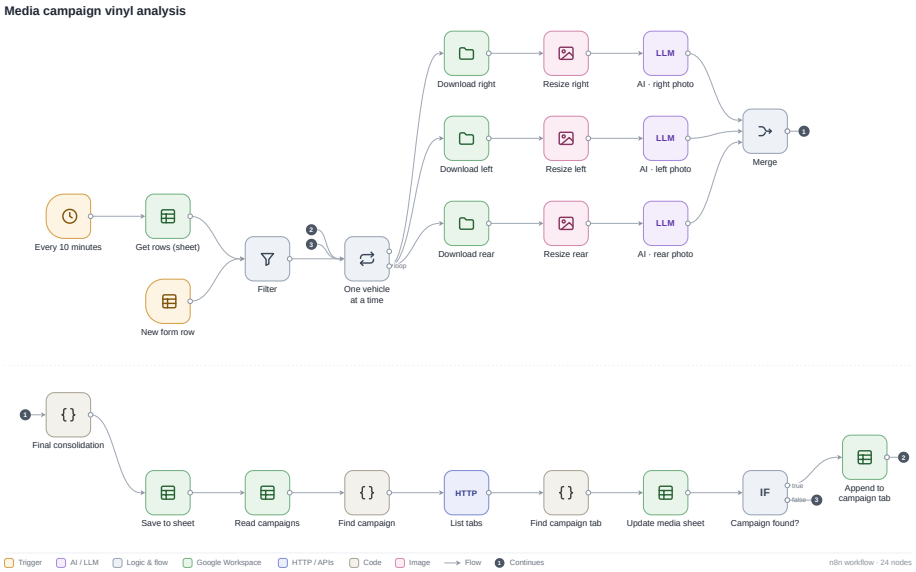
Live plate OCR `5 nodes`

Reads the plate from a photo while the user is still on the form and returns it, so the form can warn about a mismatch. It never blocks a submission.



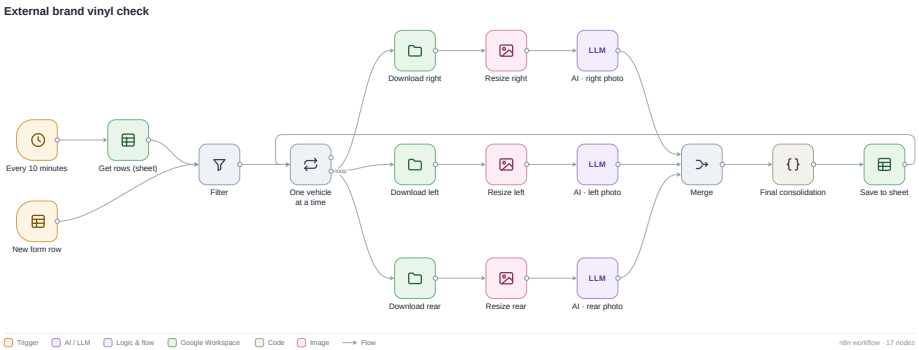
Media campaign vinyl analysis `24 nodes`

For cars wrapped for advertising campaigns: analyses the photos, matches the car to the campaign that pays for it and appends it to that campaign's tab.



External brand vinyl check `17 nodes`

Checks vinyl from external brands: brand, state, logo and plate, for cars outside the standard designs.



SECTION

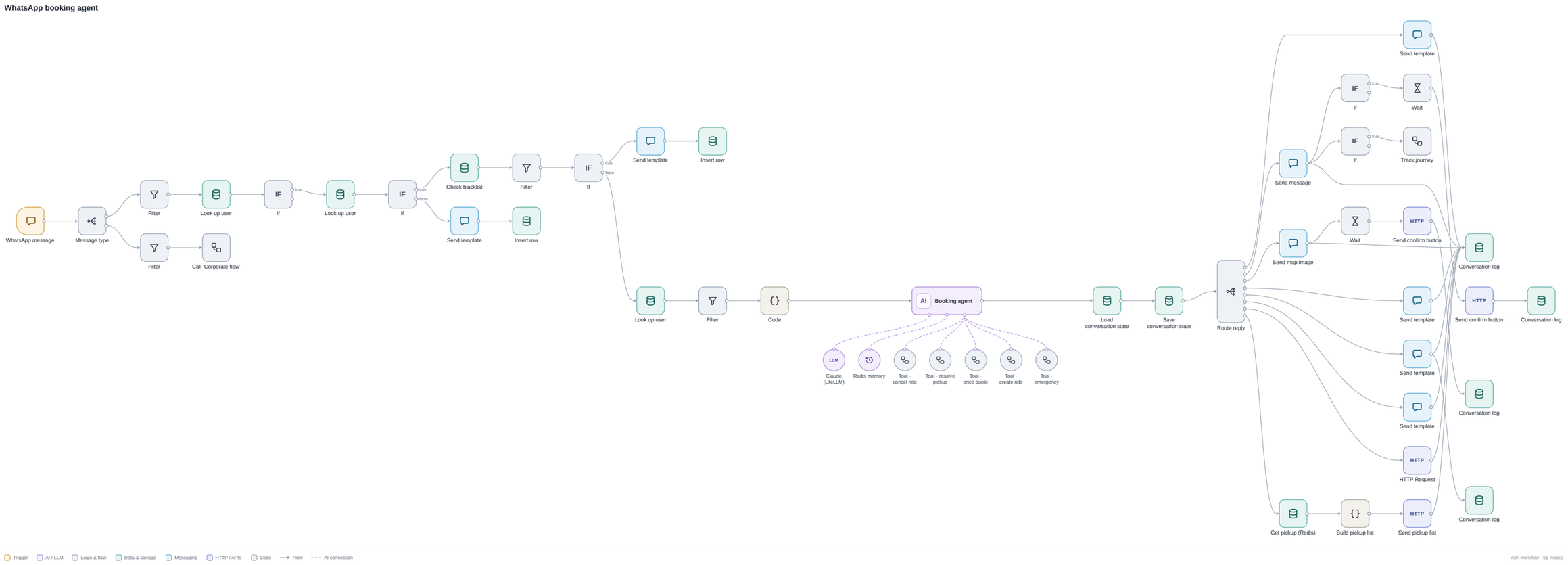
WhatsApp Ride-Booking AI Agent

Book a ride on WhatsApp without the app. A Claude agent runs the conversation in six gated phases; tools are separate workflows; state lives in Redis and a data table, and replies are chosen from fixed templates by markers in the model output.

Full case study: mathieutellene.github.io/work/whatsapp.html

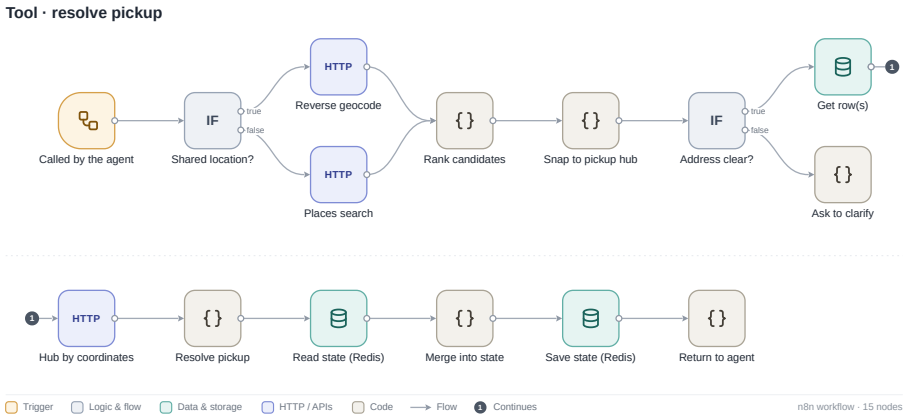
WhatsApp booking agent `51 nodes`

Entry point. Message-type switch, blacklist and flood filters, then the agent with its memory and five tools. After the agent, a router reads the marker in its output and sends one of seven reply formats (buttons, lists, map image, text) and logs the conversation.



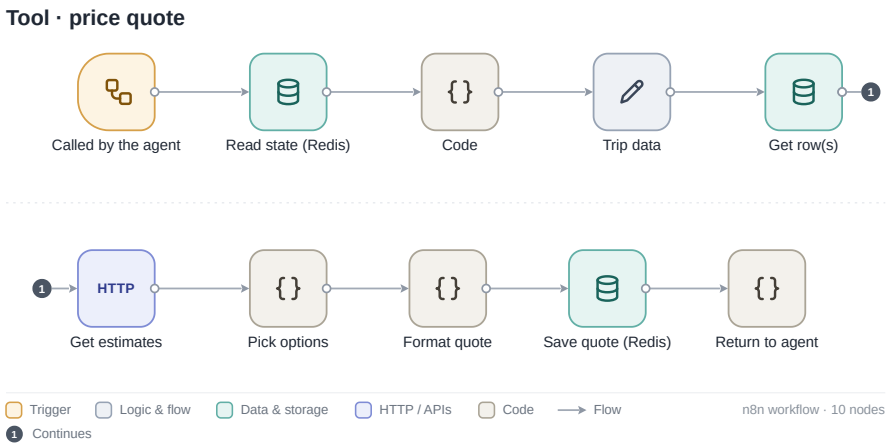
Tool · resolve pickup `15 nodes`

Tool: resolve the pickup. Chooses reverse geocoding or a places search depending on what the rider sent, ranks candidates, snaps to the nearest allowed pickup point and stores the result in the session.



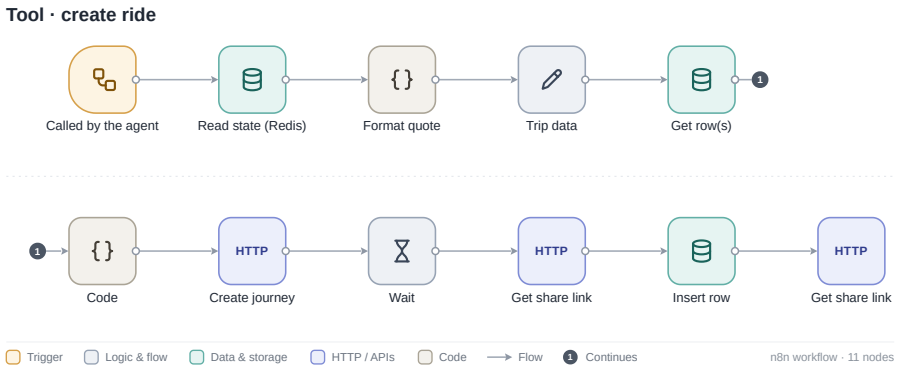
Tool · price quote `10 nodes`

Tool: get price estimates for the trip and keep the chosen option in the session.



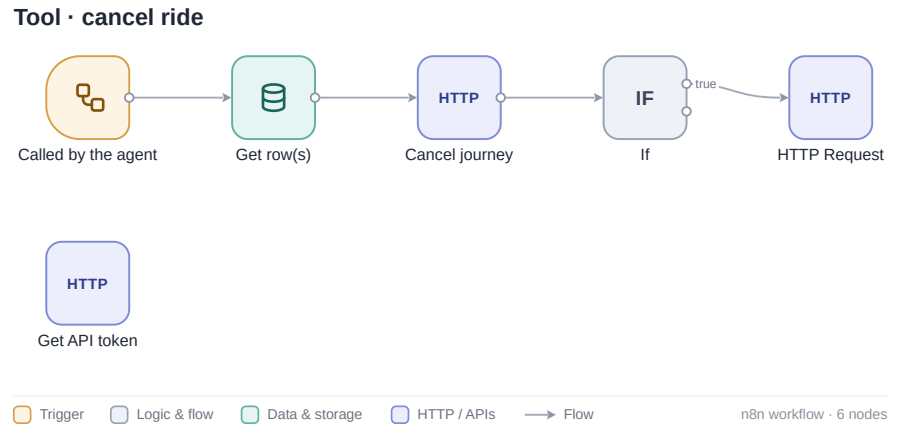
Tool · create ride `11 nodes`

Tool: create the ride through the API and return a share link.



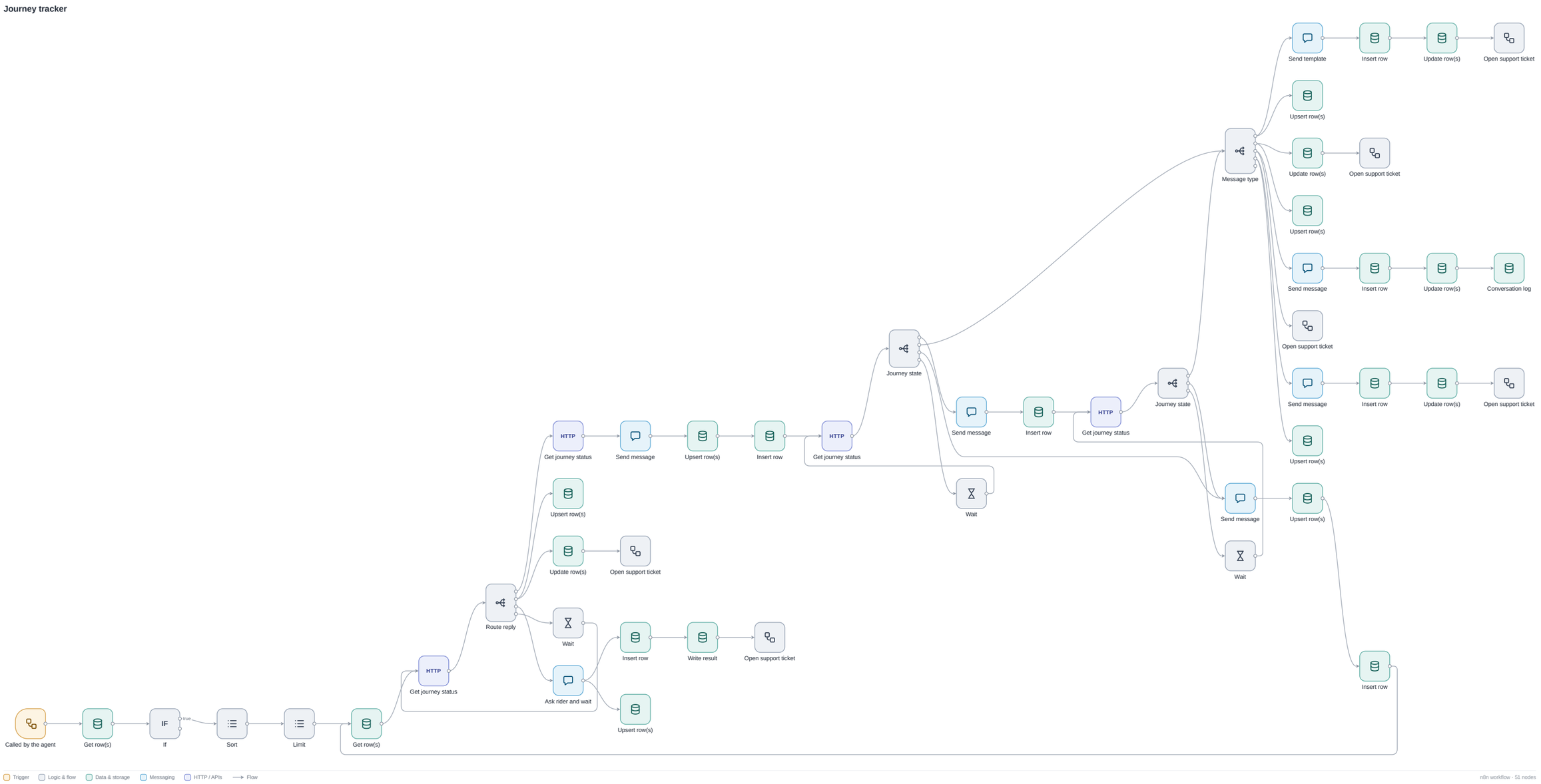
Tool · cancel ride `6 nodes`

Tool: cancel the ride. The free-cancellation window is computed in code, not by the model.



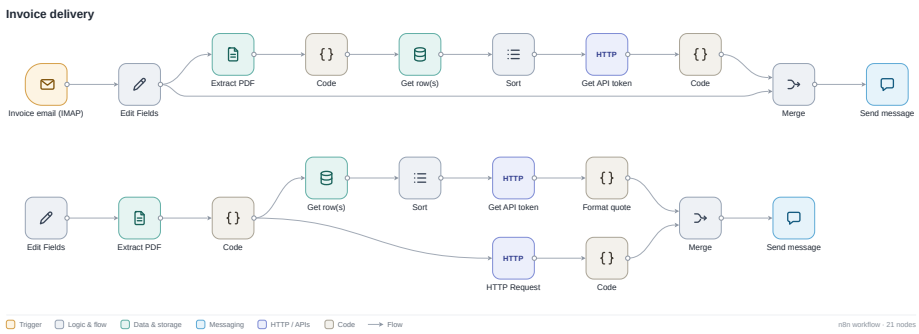
Journey tracker `51 nodes`

Tracks the ride after booking: polls its status, sends a message at each milestone, asks the rider when something looks wrong and opens a support ticket when needed.



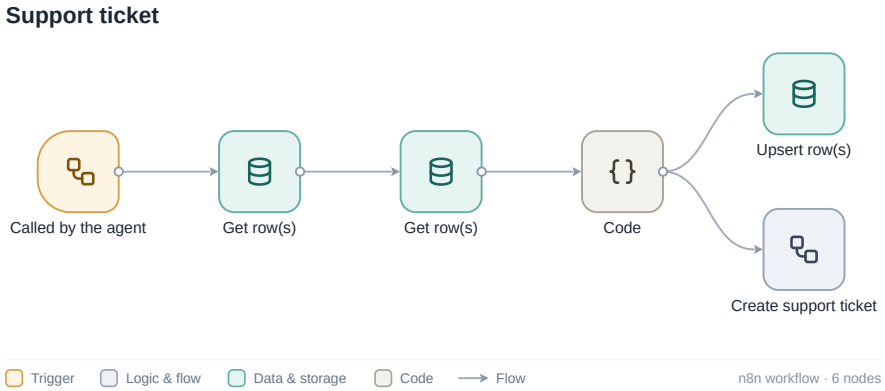
Invoice delivery 21 nodes

Reads the invoice email, extracts the PDF and sends it to the rider on WhatsApp.



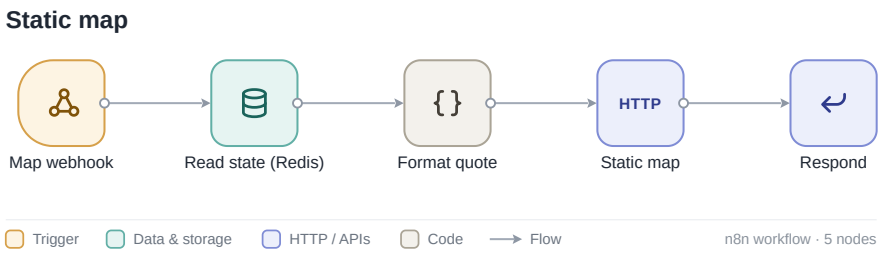
Support ticket 6 nodes

Opens a support ticket with the conversation attached.



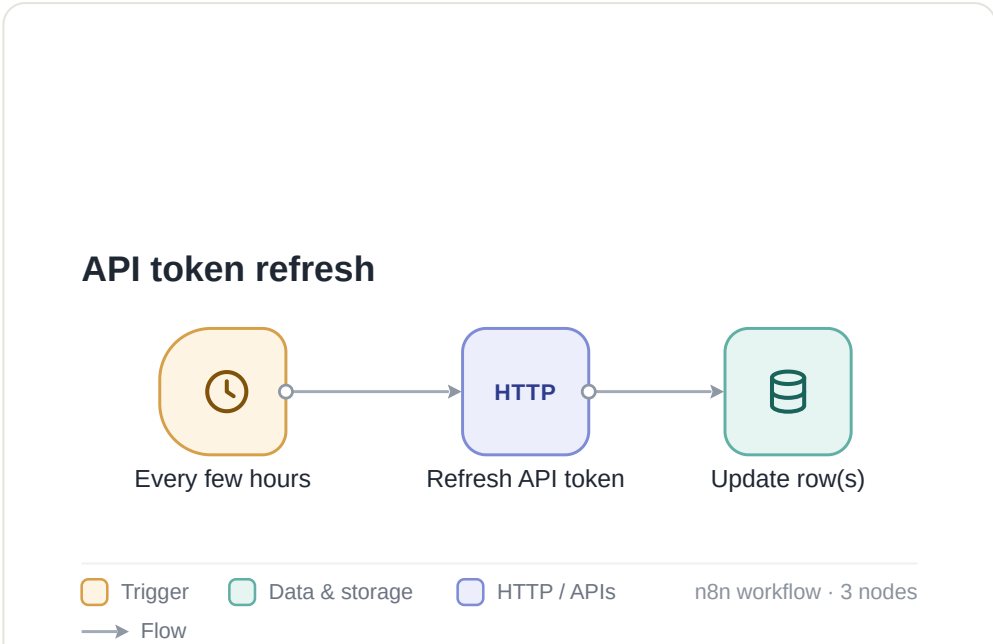
Static map 5 nodes

Returns a static map image of the pickup point for the WhatsApp message.



API token refresh 3 nodes

Keeps the API token fresh on a schedule.



SECTION

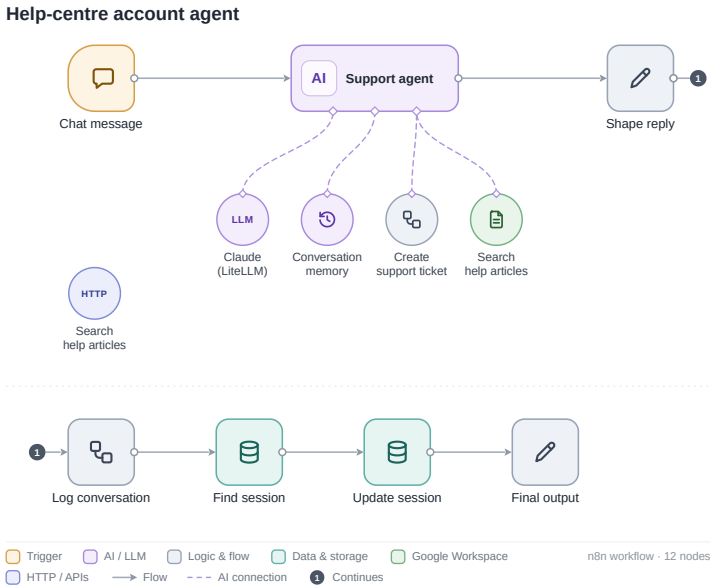
CabiBot · Help Center AI Agent

A chat agent in the rider Help Center for sign-in and account problems. It searches help articles, keeps memory per session and opens a ticket only when it cannot solve the case. 11,818 conversations so far.

Full case study: mathieutellene.github.io/work/cabibot.html

CabiBot · Help Center agent `12 nodes`

Chat trigger, the agent with its model, memory, article-search tool and ticket tool, then the reply is shaped and the session logged.



SECTION

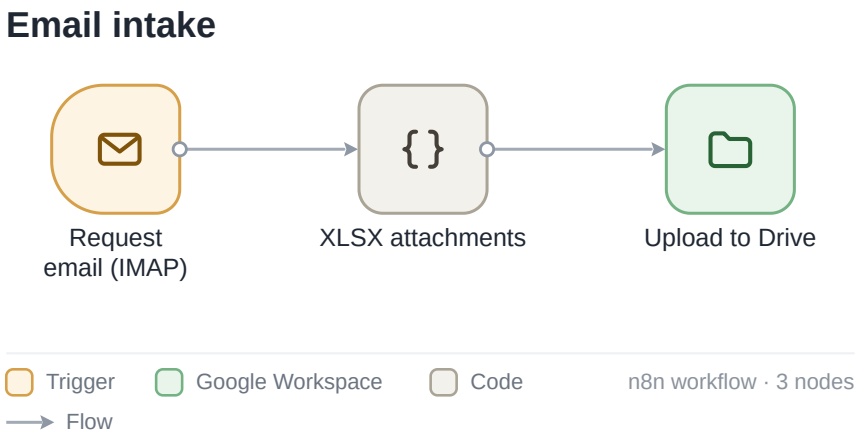
Email-to-Reservation Pipeline (B2B)

A corporate client sends trip requests as spreadsheets and the fleet partner sends signed work orders as PDFs. Two agents read both, a sheet cross-checks them, and a person creates or cancels each ride with one click; the ride goes out through the public API with its driver pre-assigned.

Full case study: mathieutellene.github.io/work/b2b.html

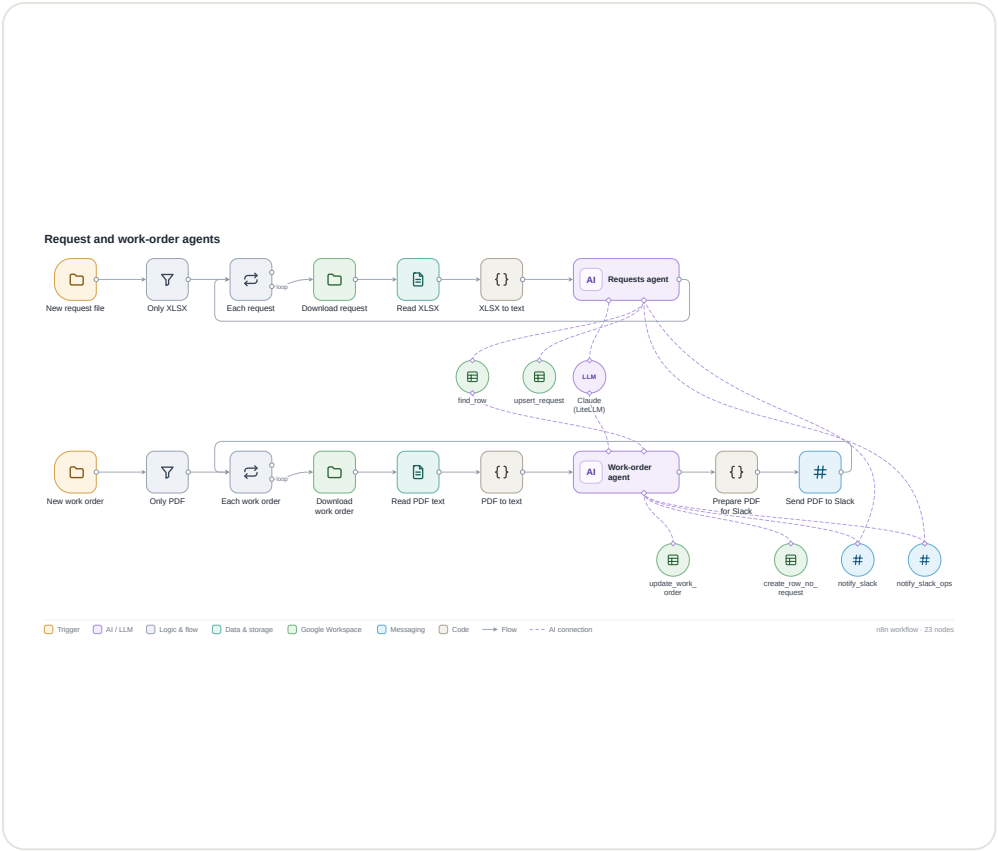
Email intake 3 nodes

Saves the client's request attachments from the mailbox to Drive.



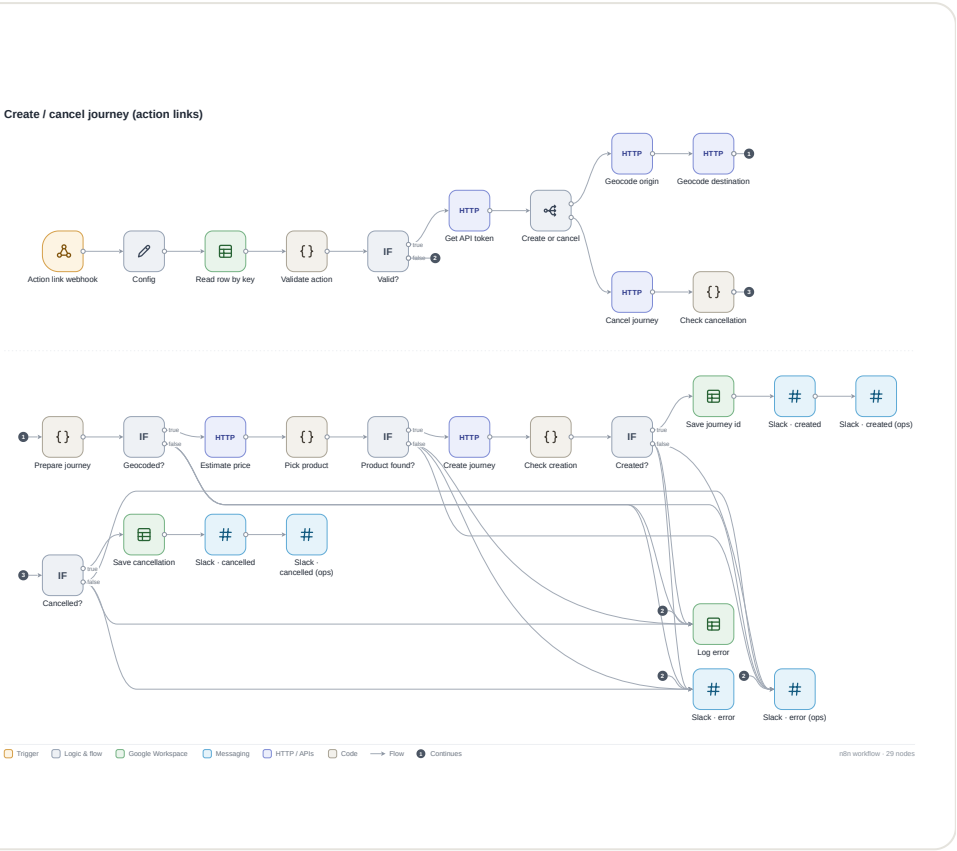
Request and work-order agents 23 nodes

Two agents, one for requests and one for work orders. Their tools can only find rows, upsert values and notify Slack; formula columns and ride fields are out of their reach.



Create / cancel ride (action links) 29 nodes

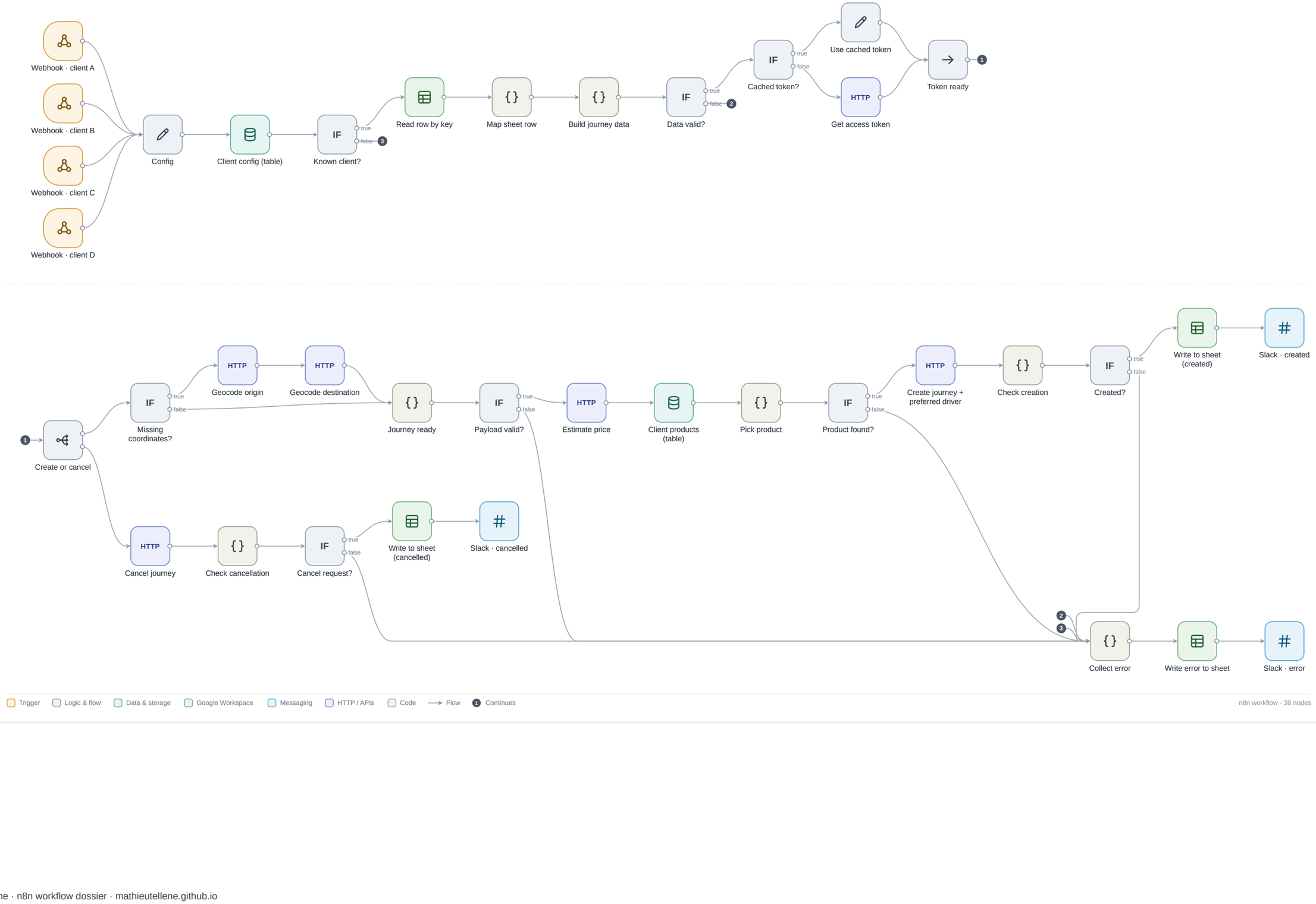
The action behind each row's create and cancel links. It re-validates on the server, authenticates, geocodes, estimates the price, picks the product, calls the API, verifies the result, writes it back and notifies.



Multi-client ride engine `38 nodes`

The same logic generalised for several clients: one webhook per client, per-client configuration and products in data tables, one error collector.

Multi-client journey engine



SECTION

COPs Control · version 1

The first version of COPs Control read dashboards as images. It was replaced by a Python and PostgreSQL pipeline on the team's Mac mini that reads the data itself, because image extraction occasionally invented a digit.

Full case study: mathieutellene.github.io/work/cops-control.html

---

COPs Control v1 · weekly KPI extraction from dashboards `92 nodes`

Ten dashboards in parallel. Each branch signs in to the BI tool, finds the workbook and view, renders it as an image, sends it to a vision model, flattens the JSON, maps it to the weekly sheet and writes it. A report that arrives by email is parsed as PDF.

SECTION

Ops orchestrator (prototype)

A prototype, not in production: one agent reachable from Slack, chat, email, a schedule and a webhook, with tools for files, email, calendar, the BI tool and tasks.

Ops orchestrator (prototype) 27 nodes

Every input is normalised to one shape, a classifier decides the intent, the agent acts with its tools and the reply is routed back to where the request came from.