CASE STUDY • AI PLATFORM • LEAD GENERATION

# AI Lead Gen Agent

A multi-agent platform that autonomously finds leads, enriches data, scores and generates personalized outreach messages

nanobot — proprietary multi-agent orchestration framework (~4,000 lines of core code)

Python 3.12 | Claude API | FastAPI | Next.js 15 | Playwright | OpenViking | Temporal | Neo4j | Docker

| 37+ | 12 | 5 | 6 phases |
|---|---|---|---|
| Tools in agent registry | Skills agent capabilities | Channels TG, WA, Web, Voice, CLI | Pipeline full lead gen cycle |

PROBLEM

# Bottleneck at the top of the B2B funnel

### Manual lead search

Managers manually search for companies on 2GIS, Rusprofile, copy contacts. 100 leads take 2-3 working days.

### No prioritization

All leads are treated equally — no scoring, no HOT/WARM/COLD segmentation. Time is wasted on non-target contacts.

### Template-based outreach

Same message for everyone — low conversion. Manual personalization is unrealistic with 500+ contacts.

### Fragmented data

Contacts in Excel, notes in Telegram, history in the manager's head. No unified database with enriched data.

ABOUT THE PRODUCT

# Overview

| | |
|---|---|
| Product | AI Lead Gen Agent (autonomous) |
| Market | B2B: construction, security, fitness (Russia) |
| Core | nanobot — ~4,000 lines, 37+ tools |
| AI | Claude OAuth / LiteLLM (multi-provider) |
| Channels | Telegram, WhatsApp, Web, Voice, CLI |

### What the agent does

Autonomously performs the full B2B lead generation cycle: semantic core collection, data scraping from sources, company enrichment, lead scoring (HOT/WARM/COLD), and personalized outreach message generation. All business logic is configured through dialogue — Zero-Hardcode.

AI PIPELINE

# 6 phases of automated lead generation

Fully autonomous pipeline from keywords to ready-made outreach messages. Real-time WebSocket progress streaming.

1 **SEMANTIC CORE**
Keyword extraction for the niche. Yandex Wordstat + Google via KDL scripts

2 **NICHE ANALYSIS**
Niche analysis: competitive landscape, market capacity, key players

3 **SCRAPING**
Data collection from 2GIS, Rusprofile, websites. Playwright + stealth. Deduplication by TIN

4 **ENRICHMENT**
Enrichment: email, phone, social media, revenue, employees. Cross-verification

5 **SCORING**
Scoring and segmentation: HOT / WARM / COLD by size, activity, ICP fit

6 **OUTREACH**
Personalized message generation based on each lead's data and pain points

MULTI-AGENT

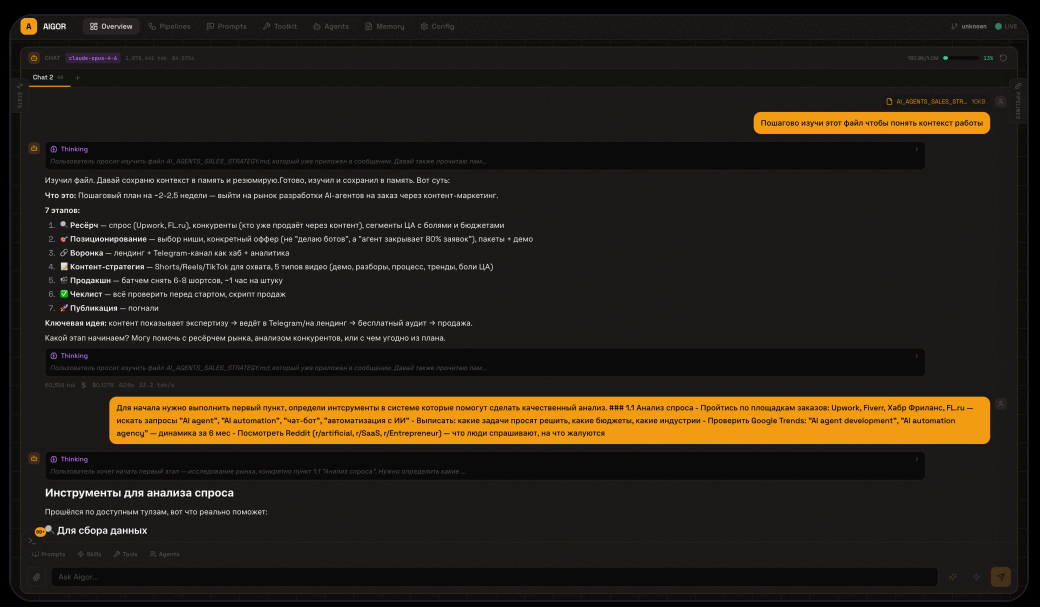
# Tool Visibility Layer

| MECHANISM | DESCRIPTION | DETAILS |
|---|---|---|
| Tool Visibility | LLM sees 12 tools, orchestrator manages 37+. Token savings and model focus | 2 levels |
| Sub-agents (spawn) | Main agent spawns sub-agents for parallel tasks: city scraping, enrichment | parallel |
| Context Compaction | 50% context — tool results compression. 75% — old messages removal | auto |
| Error Detection | Tracker blocks repeating errors after 2 identical calls | kill-switch |

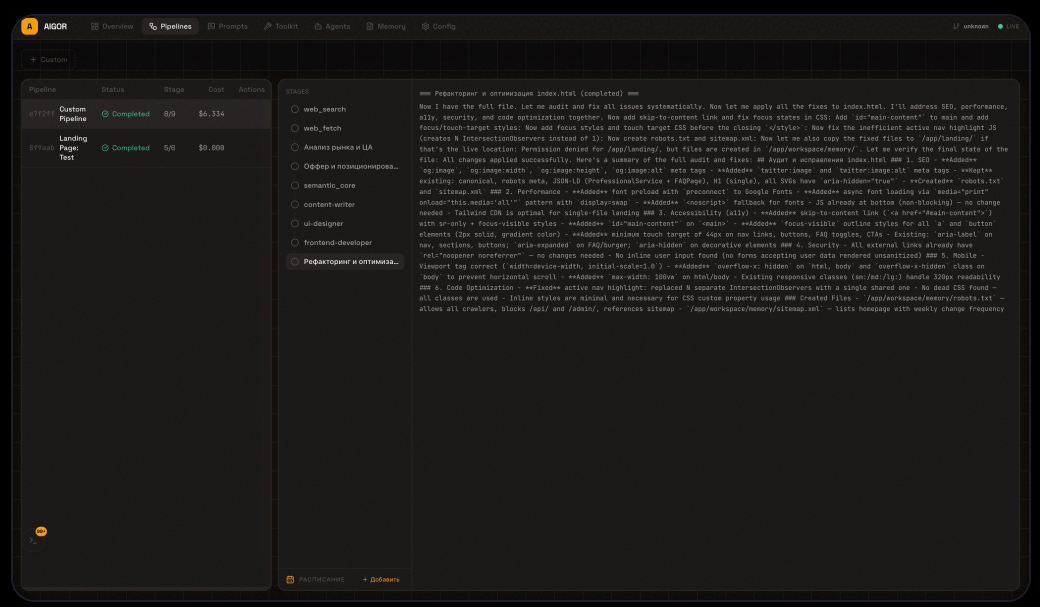
DASHBOARD

# Real-time Admin Panel

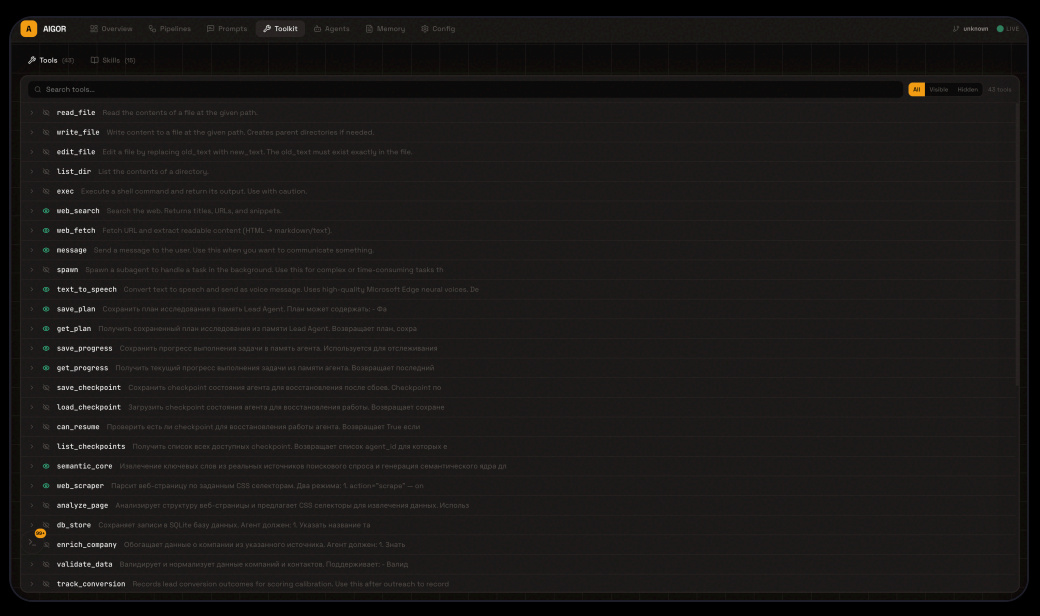
Next.js 15 dashboard with WebSocket streaming. Full control over the agent, pipelines, tools and memory:



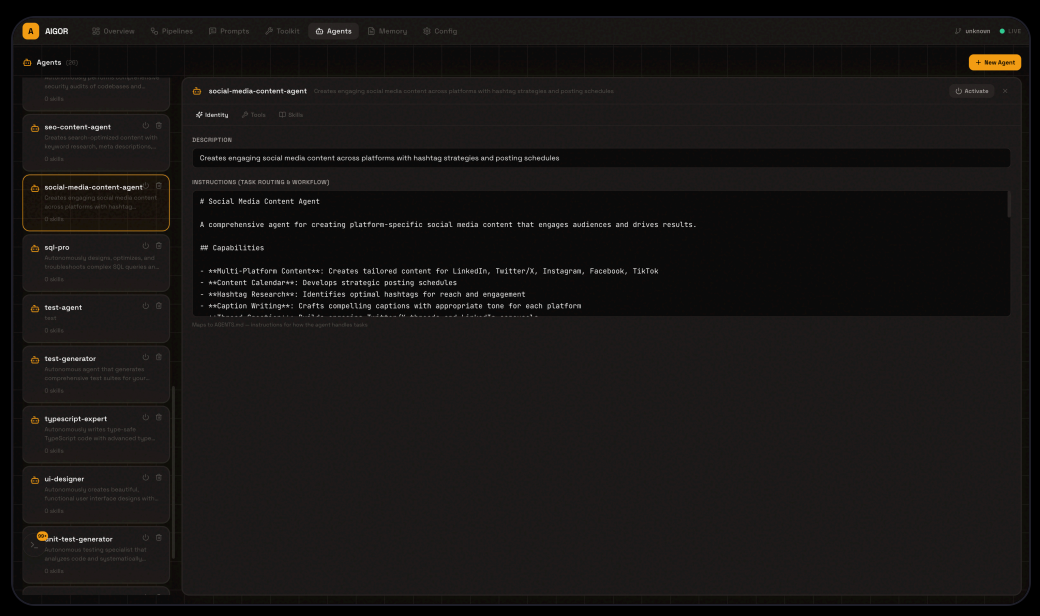

Overview — AI chat with real-time agent thought streaming




Pipelines — pipeline constructor with stages and logs



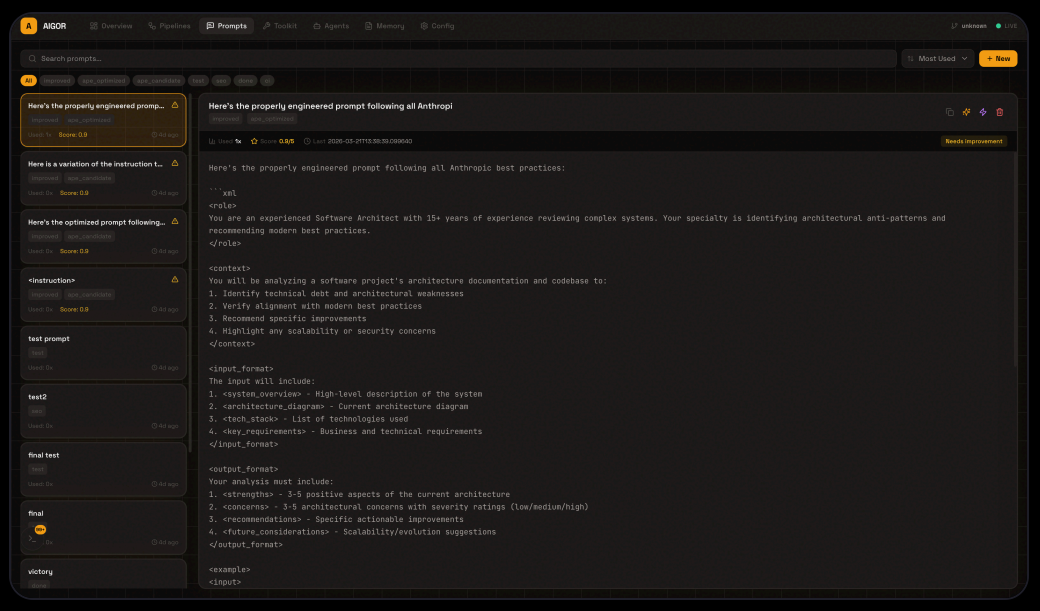

Toolkit — 37+ tools with LLM visibility flag



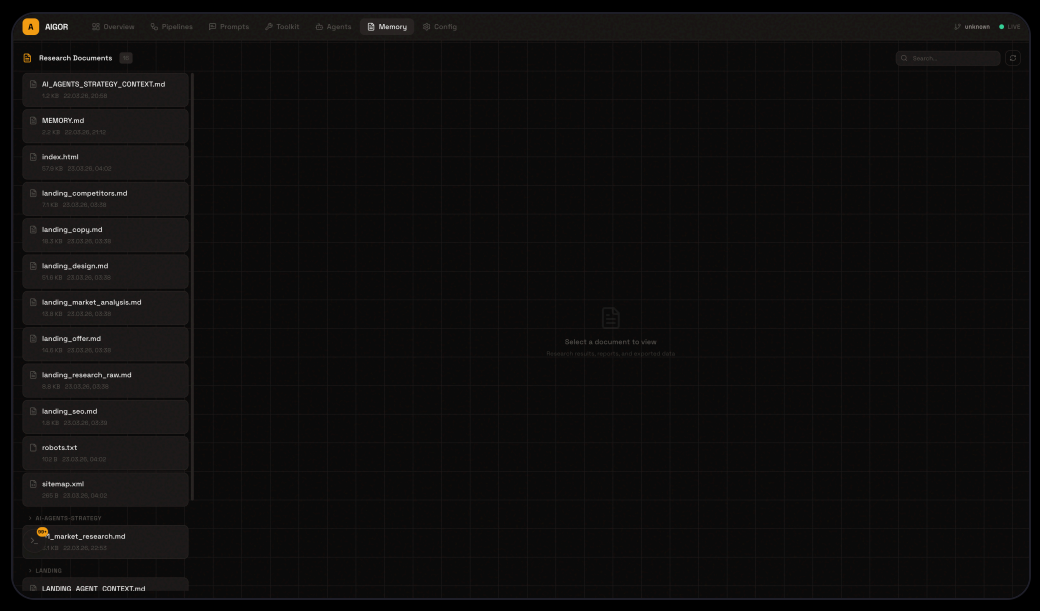

Agents — agent CRUD, tools and skills configuration

DASHBOARD

# Prompts & Memory




Prompts — prompt library with APE (auto prompt engineering)




Memory — research results and agent documents

CONTEXT MEMORY

## OpenViking — agent knowledge base

| CAPABILITY | APPLICATION IN THE PROJECT |
|---|---|
| 3-level context (L0/L1/L2) | L0 (~100 tokens) for screening, L1 (~2K) for planning, L2 (full text) on demand. Token consumption reduction up to 91% |
| Semantic Search | Pipeline stages automatically search for context via `context_query`. Results are appended to the stage prompt |
| Session Logging | All LLM inputs/outputs are logged into sessions for long-term user memory extraction |
| Resource Caching | Scraped websites and analyses are saved as resources. Repeated queries by industry are served from cache |

ZERO-HARDCODE

# Configuration through dialogue

All business logic lives in markdown files (`workspace/memory/`). Changing the niche, city, or strategy = updating a file through dialogue with the agent, no commits needed:

**SOUL.md**

Agent personality and behavior (~400 lines): tone, strategies, constraints

**memory/NICHES.md**

Target niches — populated from operator dialogue

**memory/SOURCES.md**

Data source catalog + CSS selectors for scraping

**skills/ (12 skills)**

Each skill = SKILL.md + examples + templates. Auto-pickup by ContextBuilder

**Result:** An operator switches the agent to a new market in 5 minutes via chat. Adding an industry, cities, sources — all through `write_memory`, without any code changes.

ARCHITECTURE

# Technology Stack

| COMPONENT | TECHNOLOGIES |
|---|---|
| **Agent Core** | Python 3.12, Typer, Pydantic 2, asyncio. Agent loop up to 35 iterations |
| **LLM** | Claude OAuth (Max/Pro subscription) / LiteLLM (Anthropic, OpenRouter, OpenAI, Gemini). Auto-fallback |
| **API + Dashboard** | FastAPI + WebSocket (real-time streaming). Next.js 15, React 19, Tailwind CSS 4, Framer Motion |
| **Scraping** | Playwright + stealth mode, Tadpole (KDL scripts for Yandex Wordstat / Google) |
| **Database** | SQLite (aiosqlite), OpenViking v0.2.6 (context DB: L0/L1/L2, search, sessions), Neo4j (graph) |
| **Orchestration** | Temporal (workflow engine), cron-based scheduling, Pipeline Constructor in dashboard |
| **Channels** | Telegram, WhatsApp (Node.js bridge), Web (WebSocket), Voice (TTS/OGG), CLI |
| **Monitoring** | Loguru, Sentry, OpenTelemetry — LLM call and tool execution traces in SQLite |
| **Deploy** | Docker Compose — 6 services: nanobot, openviking, temporal, neo4j, temporal-ui, dashboard |

CHANNELS

## Multichannel Communication

A unified Message Bus (async FIFO) decouples channels and the agent loop. Any channel is simply a publisher/subscriber:

**Telegram**
python-telegram-bot
primary channel

**WhatsApp**
Node.js bridge
linked via QR

**Web + Voice + CLI**
WebSocket, TTS (OGG)
interactive mode

---

TESTS

## Test Coverage

| MODULE | TESTS |
|---|---|
| Pipeline Manager (create, persistence, OpenViking, state control) | 47 |
| AI Chat (rendering, tabs, attachments, context bar) | 34 + 27 |
| Agent CRUD, context switching, tool filtering | ~30 |
| Claude OAuth Provider (token, tools, attachments, config priority) | 23 |
| Prompt Sync, APE, Semantic Core, Tool Validation, Skills | ~60 |

---

SUMMARY

## Results

**37+**
Tools
in agent registry

**12**
Skills
agent capabilities

**5**
Channels
TG, WA, Web, Voice, CLI

**~4,000**
Core lines
nanobot framework

**Key takeaway:** Full cycle from "set the niche and cities" to "ready leads with personalized messages" — autonomously, without human involvement. Switching to a new market takes 5 minutes via dialogue.