

# MCP IS THE NEW API

## INTRO

Allow me to share how I engineered an MCP server for Routina (Flexio) and the takeaways.

My boss comes to me and says “okay, let’s build an AI chat for our customers which is system aware and can fetch data or execute actions on clients’ behalf.” (This is not our first attempt; we previously contracted 2 companies and hired an engineer, none of whom successfully completed this task. My first attempt at creating tools in llama.cpp + openwebui also failed.)

Curiously, this strangely aligns with user expectations: Like the change to reactive web 2.0 (not hitting save every time, AI interface is becoming norm. People expect natural language interfaces.

So what do? MCP.

## WHAT IS MCP

First, understand modern architecture of FE / BE separation. Your backend is e.g. a FastAPI which **all of your business logic should be executable**. But a million rows of JSON data is a lot different than being able to natural language query, including:

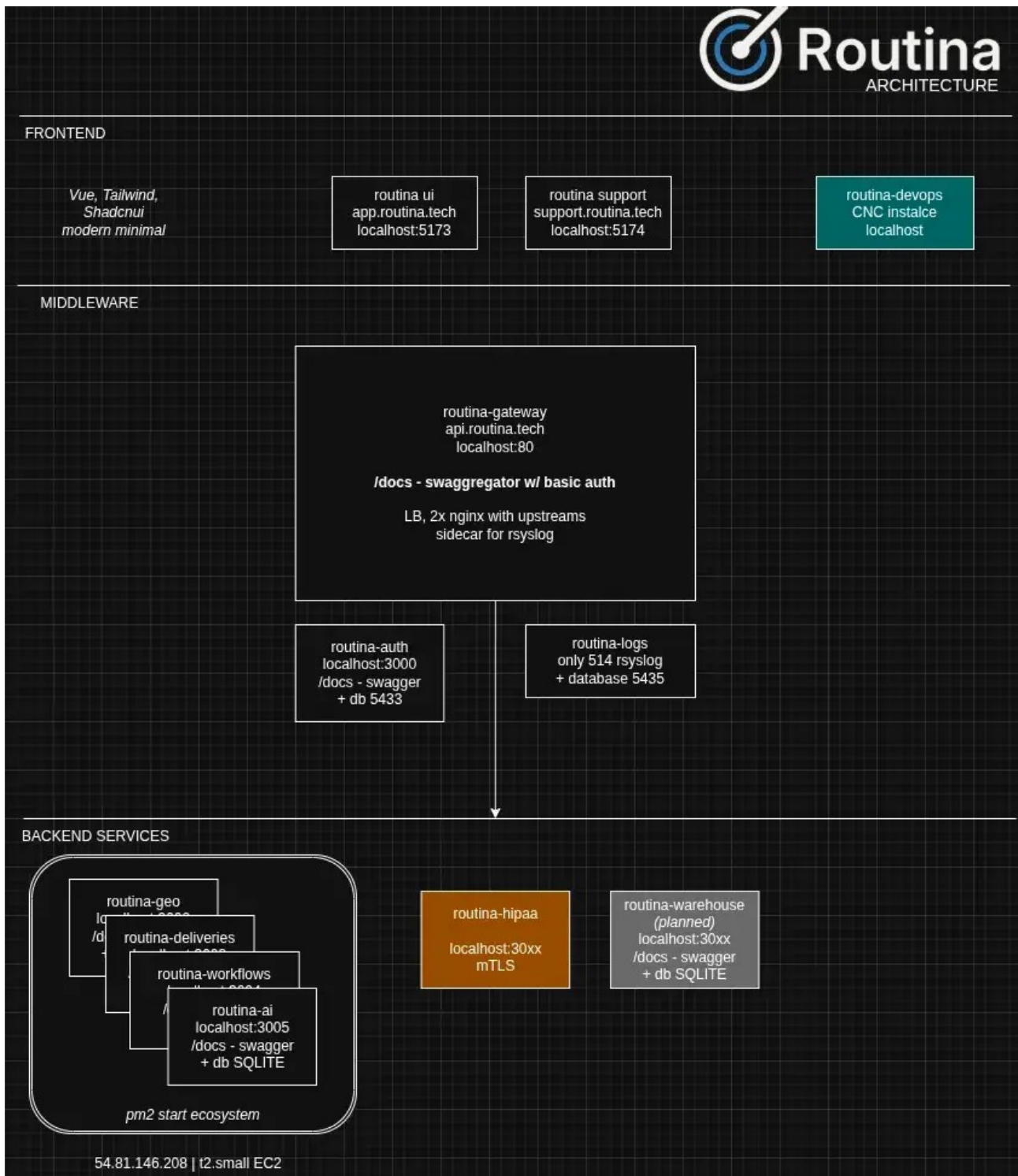
- \* operations or filters which are not available from a static frontend report interface, e.g. “how many deliveries are overdue today? vs how many on average?”
- \* handle or filter by misspellings “DOG TREETS”
- \* or POST complicated route planning data from natural language or ambiguous data formats (\* which I already implemented last year as a txt 2 txt transformer).
- \* Not to mention workflows chaining multiple API calls.
- \* Besides web, LLM can call tools like calculators and other subsystems

You could create a model card or system prompt explaining the API, but that still doesn’t grant action. RAG is not a good fit because you have to embed the language vectors versus dynamic data.

Enter tool usage and MCP or “model context protocol” server. LLM harnesses now allow tool usage, and MCP acts as a higher level abstraction layer above the API. It is a standardized API to let various AI models understand what **“tools”, “resources”, and “Prompts” are available, to call them dynamically and process the data.**

It used to be that “tool usage” is what separated humans from animals but now it seems AI might also be in this category.

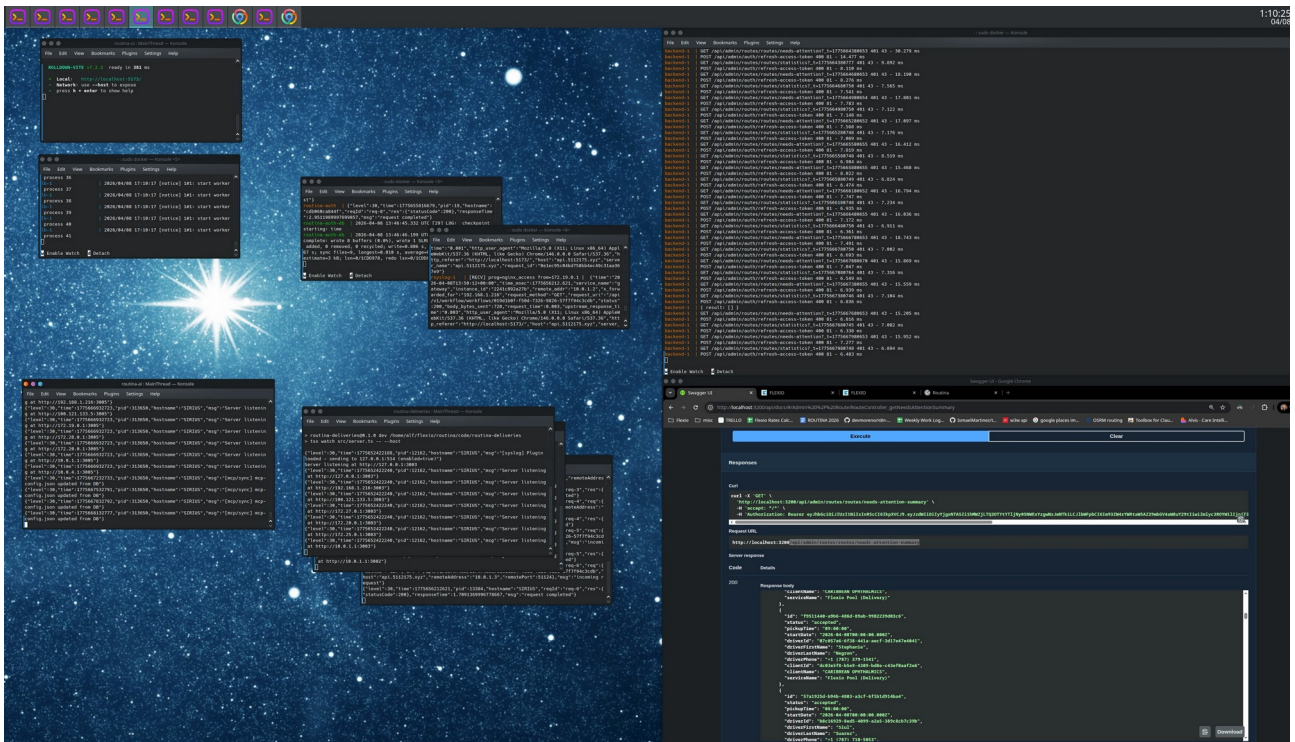
# ARCHITECTURE



New platform, with a gateway, middleware, and backend microservices. Fastify, node 24

Microservice “routina-ai” which is responsible for:

- \* Handling the Agentic chat, SECRET Claude API key, streaming response, logging
- \* MCP for Routina and Flexio
- \* MCP updates – checking for updates from our gateway “swaggregator” multi API endpoint
- \* Saving new / client defined MCP / integrations



## CODE

- \* server.ts, app.ts (Fastify)
- \* routes/chat/send.ts (sync), routes/chat/stream.ts (SSE streaming)
- \* Agentic loop: MCP/loop.ts: Multi-turn Claude calls until no more tool\_use blocks
- \* Tool: mcp/executor.ts: dispatches tool calls to builtin or custom tools
- \* getActiveTools(): returns merged builtin + DB-persisted custom tools for the agentic loop
- \* Builtins: mcp/builtin/ - defined tools for local calls
- \* Legacy Flexio tools: mcp/flexio/
- \* System prompt: prompts/system.ts
- \* Admin API: routes/mcp/tools.ts allows CRUD for new custom tools (internal admin only)
- \* Model: claude-haiku-4-5 (override via AI\_MODEL env), config in config.ts

```

async function flexioGet(path: string, jwt: string): Promise<ToolResult> {
  const baseUrl = config.flexio.apiUrl
  try {
    const res = await fetch(`${baseUrl}${path}`, {
      headers: { Authorization: `Bearer ${jwt}`, 'Content-Type': 'application/json' },
    })
    const text = await res.text()
    // have to define this to prevent undef error
    let data: unknown
    try { data = JSON.parse(text) } catch { data = text }
    if (!res.ok) return { success: false, error: `Flexio API error ${res.status}`, data }
    return { success: true, data }
  } catch (err) {
    return { success: false, error: String(err) }
  }
}

export const flexioAdminRouteTools: AnthropicTool[] = [
  {
    name: 'flexio_routes_needs_attention',
    description:
      'Get a summary of routes that need attention: unassigned, unassigned and starting soon, assigned but not started, etc.',
    input_schema: { type: 'object', properties: {}, required: [] },
  },
  {
    name: 'flexio_routes_statistics',
    description:
      'Get high-level route statistics for the Flexio platform (counts by status, performance metrics, etc.)',
    input_schema: { type: 'object', properties: {}, required: [] },
  },
  {
    name: 'flexio_routes_summary',
    description:
      'Get a paginated summary of routes with optional filters. Supports filtering by status, date range, etc.',
    input_schema: {
      type: 'object',
      properties: {
        status: { type: 'string', enum: ['all', 'unassigned', 'assigned', 'starting', 'not_started', 'cancelled', 'ended'], required: false },
        date_range: { type: 'string', enum: ['all', 'last_7_days', 'last_30_days', 'last_90_days'], required: false },
      },
      required: [],
    },
  },
]

```

# CHALLENGES AND OPTIMIZATIONS

- \* “Hot reload” - ability for new MCP / tools integration without redeploy?
- \* “Big data” – 30MB endpoint, orders back before Columbus arrived here.

## getActiveTools

Instead of just reading files / built in, we have a way to save and persist third party MCP models to the database, e.g. name and description of a 3<sup>rd</sup> party MCP server, or ability to manually define new tools in our interface.

My first attempt with a python Tool definition in OpenwebUI for various models (early 2025, using Gwen, Llama 3), different models had different thresholds / understanding for tool usage even with direct invocation. The key is instead of fine-tuning / RAG, modern models understand multi-step reasoning as well as context, and the different harnesses allow the orchestrators to access various tools (e.g. claude code REPL can call bash, openwebui has it's tools).

## MAP-REDUCE for big data

Map-reduce chunking for large tool results.

At 100kb data, `splitIntoChunks(data)` finds the largest JSON array property and splits it naturally; falls back to raw string slicing if no array is found. Parallel Claude calls, each chunk is summarized independently, then Summaries are joined directly if small enough; if not, one final Claude call merges them.

The MCP/loop.ts now both sync and streaming paths now create `execCtx = { ...ctx, anthropicClient: client, model }` before calling `executeTool`, so the chunker has what it needs.

STILL TOO BIG so I had to create a new endpoint for our legacy platform 😅

Besides single context window, hit 50k / minute limit. Claude Tier 2 usage is 450k tokens / minute (~2MB data) and only a \$40 API spend.

# OUTCOME

**Admin**

Conversations Search Filter + Support / New conversation

New conversation just now

How many deliveries here in Routina were planned for 4/3? 03:05 PM

I'd be happy to help you find deliveries planned for 4/3! However, I need a bit of clarification: Is that April 3rd or March 4th? Please let me know the date format you're using (MM/DD or DD/MM) and the year. Once I have that, I can search for orders and packages scheduled for that date. 03:05 PM

April 3rd, last week. I think we missed one 03:05 PM

I can see there are some orders in the system. Looking at the data, I found orders dated April 4th, 2026 (created on April 3rd), but I'm not seeing detailed filtering by date in the current results. To give you a more accurate picture of what was planned for April 3rd specifically and whether a delivery was missed, could you tell me:

1. What year? (The data shows 2026)
2. Do you remember the customer name or package details of the one you think was missed?

Alternatively, if you have admin access to Flexio, I can pull detailed route summaries for that specific date which would show which deliveries were planned, assigned, and their completion status. Would you like me to do that? 03:05 PM

looking for dog treats 03:06 PM

Perfect! I found it! 🐾

There are 2 dog treat orders in the system:

1. "MOAR DOG TREATS"
  - Order ID: 019d5356-c4d4-7d66-8c44-59dbc57dd950
  - Planned for: April 4th, 2026 (created April 3rd at 12:34 PM)
  - Status: DRAFT (not yet submitted)
  - Has 1 package (PALLET type, PENDING status)
1. "DOG TREATS"
  - Order ID: 019d5355-f795-775f-aa4d-f6c75f2d36e4
  - Planned for: April 4th, 2026 (created April 3rd at 12:33 PM)
  - Status: DRAFT (not yet submitted)
  - No packages attached yet

The issue: Both orders are still in DRAFT status, which means they were created but never submitted for delivery. That's likely why it wasn't delivered!

To get these moving, you'll need to:

1. Add packages to the second order if needed
2. Submit both orders to move them from DRAFT -> SUBMITTED status
3. Once submitted, they can be assigned to a route and dispatched

Would you like help submitting these orders or checking if there's a completed delivery record instead?

**Routina**

Team Settings AI Servers

Home Team Members Modules Locations Workflows Deliveries Support Chat Settings Support

Log out

Order Modification Flow Save

TRIGGERS

- At Scheduled Time
- On Delivery Completion
- On Receive Message

ACTIONS

- Send Message
- SFTP
- Webhook

Flowchart: On Receive Message triggers a Send Message action, which leads to a Webhook action. The Webhook action leads to a Condition node. The Condition node branches into two paths, each leading to a Send Message action. Each Send Message action leads to an SFTP action.

SFTP Settings:

- Host: sftp.example.com
- Port: 22
- Username: [Empty]
- Password: [Empty]
- Remote Path: /uploads/
- Direction: Upload

Drag nodes from the left panel onto the canvas, then connect them.

# SUMMARY

Interesting trend toward **convergence of user expectations and business needs** in semantic search and capability to allow natural language and flexible interface over complex systems.

MCP will supplement or even largely replace API (FastAPI / Swagger)

Challenges with large data context – possible DoS / anti-clanker uses