louzt/serpapi-mcp

Source: LOUST · **Status:** Active · **Period:** 2026-01 (active)

MCP server wrapping the SerpApi public search API: discrete tools for search, news, and image retrieval in agentic loops.

The challenge

Agentic loops need fresh web evidence to verify claims. Baking a specific search API into the agent creates lock-in; building my own bridge from scratch costs engineering time I would rather spend on the agent itself.

My role

I built a Go-based MCP server that wraps SerpApi's public search API, exposing search, news, and image search as discrete MCP tools so any agent loop can fetch fresh web evidence without hand-rolling HTTP.

What I did (3–5 decisions)

1. Discrete MCP tools (search, news, images)

Why: Each tool does one thing. An agent picks what it needs rather than parsing a single mega-response. Failure surfaces stay bounded per tool.

Tradeoff: More tools to maintain than one mega-tool. Per-tool audit clarity wins because debugging a mega-tool is where the time goes.

2. Intent-filtered broker pattern

Why: The broker surfaces the actual API call in response metadata. The agent (and the reviewer) can verify what was fetched without re-running the request.

Tradeoff: Slightly higher per-call overhead than a raw HTTP wrapper. The audit clarity earns the extra latency.

3. Read-only by default

Why: Search is read-only — there is nothing to mutate. Keeping the default read-only means a future feature that adds mutation has to go through the same opt-in ceremony as the rest of the bridge.

Tradeoff: None worth flagging. The pattern stays consistent with the other MCP bridges I ship.

What changed

Search API lock-in

Baseline: Baked into the agent loop

Swappable — SerpApi today, Bing tomorrow

Agent loop portability

Baseline: Tied to SerpApi HTTP specifics

MCP-portable across any agent framework

Evidence freshness

Baseline: Up to 24h stale (model training cutoff)

Live (real API call per retrieval)

Overall tradeoffs

I depend on SerpApi's API uptime for the bridge to work. The bridge pattern means swapping providers when needed — the agent loop stays portable through the swap.

Lessons

Agent portability comes from MCP, not from the underlying API choice. The bridge is the abstraction layer that lets the search vendor change without rewriting the agent.

Stack

Go | MCP | SerpApi

Repository

github.com/louzt/serpapi-mcp

Evidence

- [repo] github.com/louzt/serpapi-mcp