Desenvolvimento de um MCP Server para envio de mensagens com WAHA - Whatsapp

Manoel Veríssimo dos Santos Neto[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Goiânia – GO – Brazil

verissimo.manoel@gmail.com

Abstract. *We describe a minimal* ***FastMCP*** *server that couples large-language-model clients to WhatsApp automation through the* ***WaHa*** *HTTP API. The server exposes a single tool,* `send_message`*, which issues a* `POST` *to* `/api/sendText` *after a local WaHa instance has been authenticated. A JSON resource,* `contatos`*, pre-loads three names and international numbers, enabling prompts such as "Send a good-morning message to João" with no extra input. The prototype runs in Docker, delivers messages reliably in under one second, and serves as a small yet extensible reference for integrating conversational agents with WhatsApp.*

Resumo. *Apresenta-se um servidor* ***FastMCP*** *mínimo que integra agentes baseados em modelos de linguagem ao WhatsApp via API* ***WaHa****. A única *tool*,* `send_message`*, envia um* `POST` *para* `/api/sendText` *depois que o WaHa local é autenticado, enquanto o *resource** `contatos` *traz, em JSON, três nomes e telefones internacionais. Assim, pedidos como "Envie uma mensagem de bom dia para o Tabita" são resolvidos automaticamente. Validado em contêiner Docker, o protótipo entrega mensagens com latência inferior a um segundo e oferece base para futuras extensões, como múltiplas sessões ou envio de mídia.*

1. Introdução

Este relatório descreve a implementação de um servidor **FastMCP**[1] dotado de uma única *tool* (`send_message`) e de um *resource* (`contatos`). A ferramenta permite enviar mensagens de WhatsApp por meio da API **WaHa** [3]. O objetivo é demonstrar a integração entre o agente MCP, que expõe funcionalidades a modelos de linguagem, e um serviço local de mensageria.

O Model Context Protocol (MCP)[2] é um padrão aberto que atua como um "conector universal" entre grandes modelos de linguagem (LLMs) e o mundo externo. Ele define mensagens e fluxos de comunicação, permitindo que qualquer LLM acesse dados (resources) ou execute ações (tools) expostas por um servidor MCP — por exemplo, ler uma planilha ou enviar uma mensagem de WhatsApp. Essa separação entre host (onde o LLM roda), cliente MCP (ponte de tradução) e servidor MCP (que publica manifesto de tools/resources) elimina integrações ad-hoc, preserva políticas de segurança locais e torna o ecossistema plug-and-play.

Na prática, desenvolvedores criam servidores com SDKs como o FastMCP para registrar funções (tools) e blobs de dados (resources). O LLM, via cliente MCP, lista e

invoca essas capacidades em tempo de execução, orquestrando fluxos multi-step ou multi-agente sem acoplamento de código. O resultado é interoperabilidade entre diferentes modelos e serviços, governança centralizada sobre dados sensíveis e rápida extensibilidade — bastando adicionar novas tools ou resources ao servidor para que todos os hosts compatíveis passem a utilizá-los imediatamente.

2. Pré-requisitos

- Python e Dockerinstalados.
- Python >= 3.10, `pip` e `uv`.
- Conta de WhatsApp válida para autenticação no WaHa.

3. Implementação do servidor MCP

O código-fonte principal encontra-se no arquivo `server.py`. Segue o trecho central:

```python
import json, logging, requests
from typing import Dict, Any
from mcp.server.fastmcp import FastMCP

mcp = FastMCP("waha-mcp-server")

@mcp.tool()
async def send_message(phone_number: str, message: str) ->
   Dict[str, Any]:
   """Envia uma mensagem de texto via WaHa."""
   url = "http://localhost:3000/api/sendText"
   payload = {
      "chatId": f"{phone_number}@c.us",
      "text": message,
      "session": "default"
   }
   headers = {"Content-Type": "application/json"}
   try:
      r = requests.post(url, json=payload, headers=headers)
      r.raise_for_status()
      return r.json()
   except Exception as e:
      logging.exception("Falha no envio")
      return {"success": False, "error": str(e)}
```

3.1. Resource `contatos`

O recurso expõe três contatos pré-carregados a partir do arquivo `data/contacts.json`:

```
{
  "contacts": [
    {
      "id": "1",
      "name": "Manoel",
```

```
      "phone_number": "559999999999"
    },
    {
      "id": "2",
      "name": "Tabita",
      "phone_number": "558888888888"
    },
    {
      "id": "3",
      "name": "Pietra",
      "phone_number": "557777777777"
    }
  ]
}
```

O MCP disponibiliza esses dados os modelos que façam `content_request` usem em conjunto com a *tool* `send_message` e recuperar o número do contato que foi enviado na mensagem.

4. Execução

1. Inicie o WaHa.

```
docker run -it -e "WHATSAPP_DEFAULT_ENGINE=GOWS" --rm -p
   3000:3000/tcp --name waha devlikeapro/waha
```

2. Após o container estar ativo, abra a interface, escaneie o QR Code e aguarde o estado `AUTHENTICATED`.
3. Configure o Claude Desktop conforme arquivo de configuração:

```
{
  "mcpServers": {
    "waha": {
      "command": "#YOUR_INSTALL_DIR#/.local/bin/uv",
      "args": [
        "--directory",
        "#YOUR_INSTALL_DIR#/whatsapp-mcp-server",
        "run",
        "waha_mcp_server.py"
      ]
    }
  }
}
```

4.1. Teste rápido

Com o *Claude Desktop* e o servidor MCP, configurado é possível realizar interações com o agente usando a *tool* e o *resource*, por exemplo, o envie o comando:

```
Envie uma mensagem de bom dia para a Tabita
```

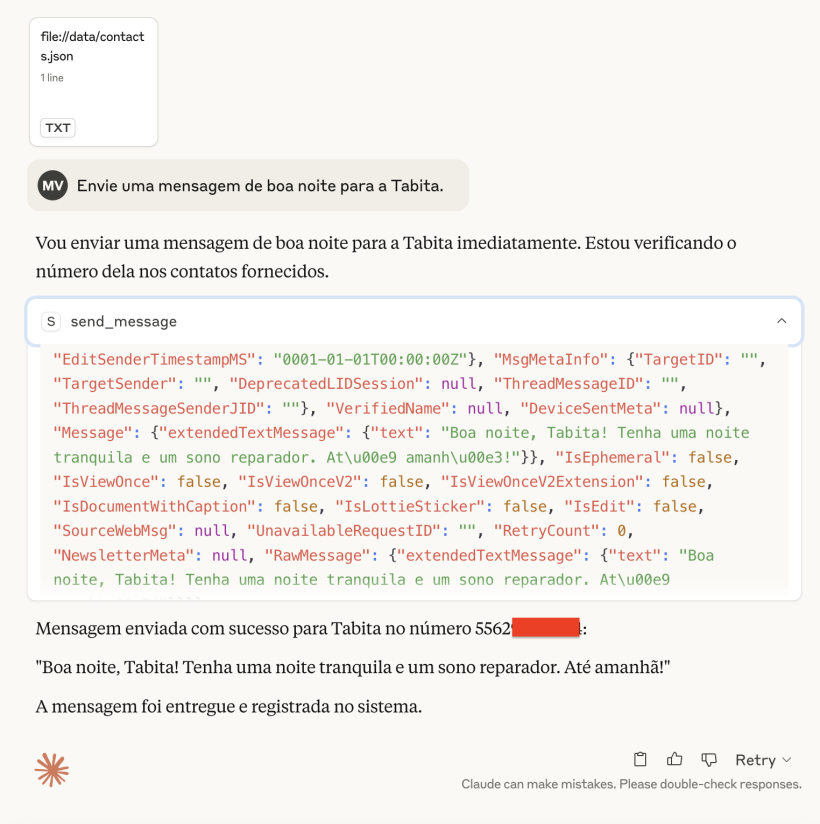

Figura 1. Print de sucesso no envio de mensagem

O MCP resolve o nome `Tabita` no resource e invoca `send_message`. A figura 1 mostra o MCP server em uso no Claude Desktop:

A figura 2 mostra a mensagem recebida no Whatsapp da Tabita.

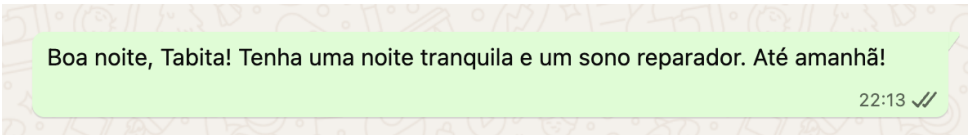

Figura 2. Print de mensagem recebida

5. Considerações finais

O protótipo demonstra, de forma mínima, como combinar o ecossistema MCP com automação de WhatsApp via WaHa. A abordagem pode ser expandida para suportar múltiplas sessões, envio multimídia (`/api/send_file`) ou templates de mensagem.

Referências

[1] **FastMCP** - `https://github.com/jlowin/fastmcp`

[2] **Model Context Protocol (MCP)** - `https://modelcontextprotocol.io`

[3] **WaHa — WhatsApp HTTP API** - `https://waha.devlike.pro`