

Universidad del Valle de Guatemala

Facultad de ingeniería

Redes

Catedrático: Jorge Yass

## **Proyecto MCP**

Diego Linares 221256

## Repositorios

- Cliente y Host: <https://github.com/DiegoLinares11/MCP-Host>
- Servidor Remoto (Supabase Admin Helper):  
<https://github.com/DiegoLinares11/MCP-Host/tree/main/src/remoteMCP>
- MCP local: <https://github.com/DiegoLinares11/Proyecto1-MCP>
- Demostración de funcionamiento: <https://youtu.be/UbPiTl54kO4>

## MCP Local – Host

El MCP Local implementado funciona como cliente principal que orquesta:

- Herramientas de FileSystem y Git.
- Herramientas de Supabase Admin Helper como crear usuarios, enviar magic links, listar usuarios, actualizar metadata, etc.
- Herramientas de SQLScout para explicar, diagnosticar y optimizar consultas SQL.

El host expone comandos de consola como:

- :tools [FS|Git|SQLScout]
- :load <ruta.sql>
- :explain <SQL>
- :diagnose <SQL>
- :optimize <SQL>
- :apply <DDL>

De esta manera, el host sirve como interfaz natural en lenguaje humano para invocar distintas operaciones.

## MCP Remoto – Supabase Admin Helper

El servidor remoto está hecho en Flask (Python) y se ejecuta en el puerto 5000.  
Expone herramientas vía JSON-RPC:

Herramienta	Parámetros	Descripción
create_user	email, password	Crea un usuario en Supabase.
list_users	–	Lista todos los usuarios registrados.

get_user_by_id	user_id	Recupera la información completa de un usuario.
update_user_metadata	user_id, user_metadata	Actualiza metadata del usuario.
reset_user_password	email	Envía correo de reseteo de contraseña.
send_magic_link	email	Envía enlace mágico de autenticación.
delete_user	user_id	Elimina un usuario de manera irreversible.
get_user_stats	rango	Estadísticas de usuarios (día, semana, mes).
bulk_invite_users	emails[]	Invita múltiples usuarios con magic link.

Análisis de Wireshark

Adapter for loopback traffic capture

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

Time	Source	Destination	Protocol	Info
449	2025-09-23 18:21:55.6...	127.0.0.1	HTTP	GET /health HTTP/1.1
453	2025-09-23 18:21:55.6...	127.0.0.1	HTTP	HTTP/1.1 200 OK
524	2025-09-23 18:21:59.4...	127.0.0.1	HTTP	POST /mcp HTTP/1.1
528	2025-09-23 18:21:59.4...	127.0.0.1	HTTP	HTTP/1.1 200 OK

Hypertext Transfer Protocol

JavaScript Object Notation: application/json

Object

Member: jsonrpc

[Path with value: /jsonrpc:2.0]

[Member with value: jsonrpc:2.0]

String value: 2.0

Key: jsonrpc

[Path: /jsonrpc]

Member: id

[Path with value: /id:1]

[Member with value: id:1]

String value: 1

Key: id

[Path: /id]

Member: method

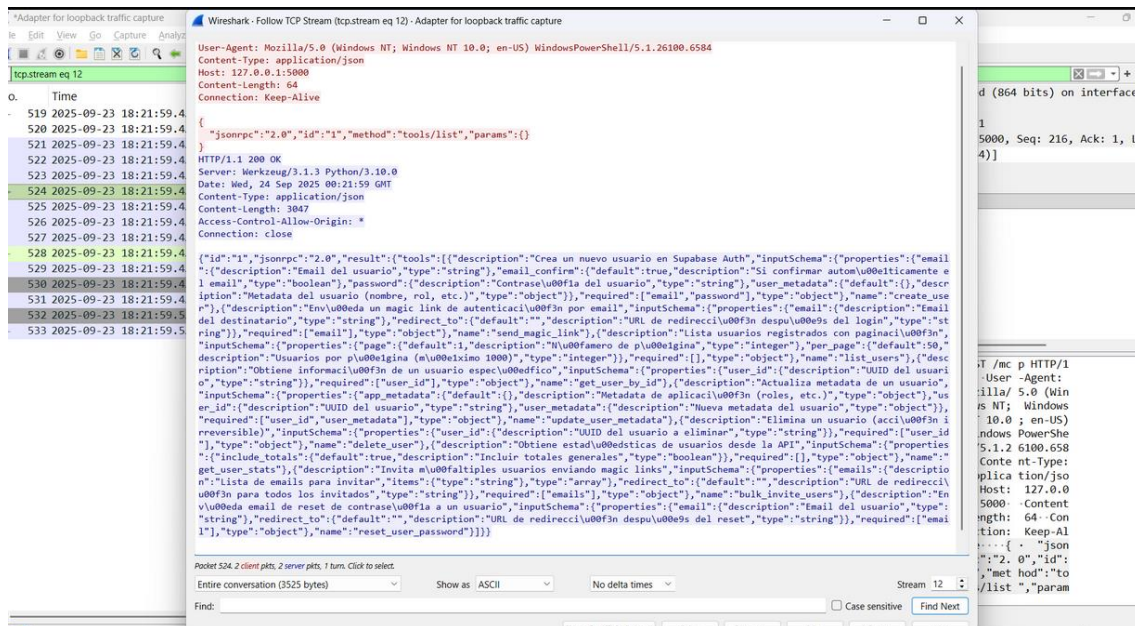
0010	2e 31 0d 0a 55 73 65 72	2d 41 67 65 6e 74 3a 20	.1-User-Agent:
0020	4d 6f 7a 69 6c 6c 61 2f	35 2e 30 20 28 57 69 6e	Mozilla/ 5.0 (Win
0030	64 6f 77 73 20 4e 54 3b	20 57 69 6e 64 6f 77 73	dows NT; Windows
0040	20 4e 54 20 31 30 2e 30	3b 20 65 6e 2d 55 53 29	NT 10.0 ; en-US)
0050	20 57 69 6e 64 6f 77 73	50 6f 77 65 72 53 68 65	Windows PowerShe
0060	6c 6c 2f 35 2e 31 2e 32	36 31 30 30 2e 36 35 38	ll/5.1.2 6100.658
0070	34 0d 0a 43 6f 6e 74 65	6e 74 2d 54 79 70 65 3a	4-Content-Type:
0080	20 61 70 70 6c 69 63 61	74 69 6f 6e 2f 6a 73 6f	application/jso
0090	6e 0d 0a 48 6f 73 74 3a	20 31 32 37 2e 30 2e 30	n-Host: 127.0.0
00a0	2e 31 3a 35 30 30 0d 0a	43 6f 6e 74 65 6e 74	.1:5000-Content
00b0	2d 4c 65 6e 67 74 68 3a	20 36 34 0d 0a 43 6f 6e	-Length: 64-Con
00c0	6e 65 63 74 69 6f 6e 3a	20 4b 65 65 70 2d 41 6c	nection: Keep-Al
00d0	69 76 65 0d 0a 0d 0a 7b	0a 20 20 22 6a 73 6f 6e	ive-...{ - "json
00e0	72 70 63 22 3a 22 32 2e	30 22 2c 22 69 64 22 3a	rpc":"2. 0", "id":
00f0	22 31 22 2c 22 6d 65 74	68 6f 64 22 3a 22 74 6f	"1", "method": "to
0100	6f 6c 73 2f 6c 69 73 74	22 2c 22 70 61 72 61 6e	ols/list ", "para
0110	73 22 3a 7b 7d 0a 7d		{} } }

JSON object (json.object), 64 bytes

Frame (108 bytes) Reassembled TCP (279 bytes)

Packets: 637 · Displayed: 4 (0.6%) · Dropped: 0 (0.0%)

Profile: Diego Linan



1. Paquetes de inicialización:
  - a. El cliente envía un POST /mcp con { "method": "tools/list" }.
  - b. El servidor remoto responde con la lista de herramientas disponibles en Supabase.
2. Paquetes de solicitud:
  - a. Ejemplo: update\_user\_metadata enviado como JSON con parámetros { "phone": "3046" }.
  - b. Visible en Wireshark con contenido JSON en claro (porque corre en localhost:5000 sin TLS).
3. Paquetes de respuesta:
  - a. El servidor responde con la metadata actualizada (ej. { "phone": "3046" }).
  - b. Esto se visualizó en Follow TCP Stream, donde aparece la conversación completa request-response.
4. Flujo de capas:
  - a. Aplicación: JSON-RPC sobre HTTP.
  - b. Transporte: TCP puerto 5000 con ACKs y segmentación.
  - c. Enlace: direccionamiento MAC en la interfaz loopback virtual.
  - d. Física: tráfico representado en software, normalmente señales eléctricas en un cable.

## Conclusiones y comentarios

- El MCP Host permite interactuar en lenguaje natural con Supabase, Git y FileSystem sin necesidad de escribir comandos bajos.
- Con Wireshark, se comprobó cómo viajan las solicitudes JSON-RPC, confirmando la comunicación cliente-servidor.

## Sobre el uso de IA

En este proyecto se apoyó de ChatGPT para:

- Estructurar el flujo de comandos del host.
- Definir la interfaz JSON-RPC del servidor remoto.
- Guiar en la configuración y captura de paquetes en Wireshark.

## Referencias

- Model Context Protocol: <https://modelcontextprotocol.io/docs/learn/architecture>
- Supabase Auth API: <https://supabase.com/docs>
- Wireshark Filters Guide: <https://wiki.wireshark.org/DisplayFilters>
- Ejemplos MCP GitHub: <https://github.com/modelcontextprotocol>