

Informe final

*Rodrigo Mansilla 22611

Redes

Semestre 2. año 2025

Ingeniería en ciencias de la computación y tecnologías de la información

Objetivo

Permitir que un LLM:

1. **Controle Spotify** (búsqueda, recomendaciones).
2. **Genere playlists contextuales.**

Modelo MCP local

Conexión

Servidor Filesystem (basado en `@modelcontextprotocol/server-filesystem`)

- **Propósito:** Manipulación básica de archivos y directorios locales.
- **Inicialización:**

```
{ "method": "initialize", "params": {} }
```

- Al iniciar, se define un conjunto de directorios permitidos (`allowed_dirs`).
- **Endpoints / Tools disponibles:**
 1. `create_directory`
 - **Parámetros:**
 - `path: str` → ruta absoluta del directorio a crear.
 - **Respuesta:** éxito o error (ej. permisos).
 2. `write_file`
 - **Parámetros:**
 - `path: str` → ruta absoluta del archivo.
 - `content: str` → contenido completo a escribir.
 - **Precondición:** el directorio padre debe existir (se garantiza con `create_directory`).
 - **Respuesta:** confirmación de escritura.
- **Notas:**
 - El cliente `execute_plan` garantiza el orden canónico → `create_directory` → `write_file` .
 - Seguridad: restringido a los directorios explícitamente permitidos.

Creación de MCP Local

Servidor Spotify (basado en `spotipy` + `OAuth 2.0`)

- **Propósito:** Integración con la API de Spotify para búsqueda, recomendaciones, creación de playlists y control de reproducción.
- **Inicialización:**

```
{ "method": "initialize", "params": {} }
```

- El servidor carga credenciales desde `.env` (`SPOTIFY_CLIENT_ID`, `SPOTIFY_CLIENT_SECRET`, `SPOTIFY_REDIRECT_URI`).
- **Autenticación:**
 - **App Client:** acceso básico (search, audio features, recomendaciones).
 - **User OAuth (con scopes):** requerido para crear playlists, añadir canciones, reproducir música.
- **Scopes soportados:**
 - `playlist-modify-public`, `playlist-modify-private`
 - `user-read-playback-state`, `user-modify-playback-state`
 - `user-top-read`, `user-read-recently-played`, `user-read-currently-playing`, `user-read-private`
- **Endpoints / Tools disponibles:**
 1. `whoami` → estado de sesión (usuario conectado o no).
 2. `auth_begin` → genera `authorize_url` para login en Spotify.
 3. `auth_complete` → completa OAuth con `code` o `redirect_url`.
 4. `search_track`
 - `query: str`, `market?: str`, `limit?: int`
 - Devuelve `{id, name, artists[], uri, preview_url}`.
 5. `analyze_mood`
 - `prompt: str` → texto de intención.
 - Devuelve una estructura `Mood` con `valence`, `energy`, `tags`.
 6. `get_recommendations`
 - `seed_tracks?: [str]`, `mood?: str`, `energy?: float`, `valence?: float`, `danceability?: float`, `tempo?: float`, `limit?: int`.
 - Devuelve lista de canciones recomendadas.
 7. `explain_selection`
 - `tracks: List[Track]`, `context: ExplainContext`.
 - Devuelve texto explicativo de la selección.
 8. `build_playlist_from_profile`
 - `mood_prompt: str`, `name?: str`, `public?: bool`, `limit?: int`.
 - Usa top tracks/artistas del perfil para armar playlist.
 9. `create_playlist`
 - `name: str`, `description?: str`, `public?: bool`.
 - Devuelve `{playlist_id, url}`.
 10. `create_playlist_with_tracks`
 - Igual que `create_playlist` pero con lista de `track_ids`.
 11. `add_to_playlist`
 - `playlist_id: str`, `track_ids: List[str]`.
 - Devuelve `{added: int}`.
 12. `ensure_device_ready`
 - Verifica dispositivos disponibles.
 - Devuelve `{status: "no_devices"|"ready"|"transferred"|"not_premium"}`.
 13. `create_public_mix` (modo bot)
 - `mood_prompt: str`, `name?: str`, `limit?: int`.
 - Genera playlist pública sin necesidad de OAuth.

MCP REMOTO

Servidor Remoto (Google Cloud Run – FastAPI)

- **Propósito:** Implementar un servidor MCP accesible públicamente, desplegado en la nube. Su funcionalidad es **trivial**: devolver la hora actual en UTC.
- **Despliegue:**
 - Imagen Docker basada en `python:3.11-slim`.
 - Servidor con **FastAPI** y **Uvicorn** expuesto en el puerto `8080`.
 - Deploy realizado en **Google Cloud Run**, con URL pública:

```
https://mcp-remote-XXXXXXX-uc.a.run.app/mcp/jsonrpc
```

Endpoints / Tools disponibles:

1. `initialize`

- **Método:** POST `/mcp/jsonrpc`
- **Entrada:**

```
{ "jsonrpc": "2.0", "id": 1, "method": "initialize", "params": {} }
```

- **Respuesta:**

```
{ "jsonrpc": "2.0", "id": 1, "result": { "status": "ok" } }
```

- **Función:** Confirma la inicialización de la sesión.

2. `tools/call` → `current_time`

- **Método:** POST `/mcp/jsonrpc`
- **Entrada:**

```
{
  "jsonrpc": "2.0",
  "id": 2,
  "method": "tools/call",
  "params": { "name": "current_time" }
}
```

- **Respuesta:**

```
{
  "jsonrpc": "2.0",
  "id": 2,
  "result": { "result": "2025-09-23 15:45:56 UTC" }
}
```

- **Función:** Devuelve la hora actual en UTC.

Captura y análisis de tráfico

Para iniciar la comunicación entre el cliente MCP y el servidor remoto en **Google Cloud Run**, primero se establece una conexión confiable mediante el protocolo **TCP**. Este proceso se conoce como **3-way handshake** y asegura que ambos extremos estén listos para intercambiar datos.

| No. | Time | Source | Destination | Protocol | Length | Info |
|------|----------------------------|----------------|----------------|----------|--------|---|
| 3977 | 2025-09-23 18:30:05.282232 | 192.168.5.234 | 200.119.162... | TCP | 66 | 54150 → 443 [ACK] Seq=1219 Ack=5975 Win=65535 Len=0 SLE=5037 SRE=5975 |
| 3978 | 2025-09-23 18:30:05.282326 | 192.168.5.234 | 34.143.73.2 | TCP | 66 | 54184 → 443 [ACK] Seq=518 Ack=6802 Win=65535 Len=0 SLE=5649 SRE=6802 |
| 3979 | 2025-09-23 18:30:05.282363 | 192.168.5.234 | 57.144.197.33 | TCP | 66 | 54181 → 80 [ACK] Seq=1015 Ack=2059 Win=65535 Len=0 SLE=2010 SRE=2059 |
| 3980 | 2025-09-23 18:30:05.285825 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 134 | Change Cipher Spec, Application Data |
| 3981 | 2025-09-23 18:30:05.286274 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 318 | Application Data |
| 3982 | 2025-09-23 18:30:05.286361 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 141 | Application Data |
| 3983 | 2025-09-23 18:30:05.324335 | 200.119.162... | 192.168.5.234 | TCP | 284 | [TCP Spurious Retransmission] 443 → 54150 [PSH, ACK] Seq=3415 Ack=1219 Win=3... |
| 3984 | 2025-09-23 18:30:05.324383 | 192.168.5.234 | 200.119.162... | TCP | 66 | [TCP Dup ACK 3977#1] 54150 → 443 [ACK] Seq=1219 Ack=5975 Win=65535 Len=0 SLE... |
| 3985 | 2025-09-23 18:30:05.380183 | 34.143.73.2 | 192.168.5.234 | TCP | 54 | 443 → 54184 [ACK] Seq=6802 Ack=949 Win=65535 Len=0 |
| 3986 | 2025-09-23 18:30:05.380183 | 57.144.197.33 | 192.168.5.234 | TCP | 66 | [TCP Dup ACK 3964#1] 80 → 54181 [ACK] Seq=2059 Ack=1015 Win=65535 Len=0 SLE=... |
| 3987 | 2025-09-23 18:30:05.441514 | 34.143.73.2 | 192.168.5.234 | TLSv1.3 | 974 | Application Data, Application Data |
| 3988 | 2025-09-23 18:30:05.443562 | 192.168.5.234 | 34.143.73.2 | TCP | 54 | 54184 → 443 [FIN, ACK] Seq=949 Ack=7722 Win=64615 Len=0 |
| 3989 | 2025-09-23 18:30:05.462262 | 192.168.5.234 | 34.143.73.2 | TCP | 62 | 54185 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM |
| 3990 | 2025-09-23 18:30:05.478554 | 34.143.73.2 | 192.168.5.234 | TCP | 54 | 443 → 54184 [FIN, ACK] Seq=7722 Ack=950 Win=65535 Len=0 |
| 3991 | 2025-09-23 18:30:05.478635 | 192.168.5.234 | 34.143.73.2 | TCP | 54 | 54184 → 443 [ACK] Seq=950 Ack=7723 Win=64615 Len=0 |
| 3992 | 2025-09-23 18:30:05.496934 | 34.143.73.2 | 192.168.5.234 | TCP | 62 | 443 → 54185 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM |
| 3993 | 2025-09-23 18:30:05.497043 | 192.168.5.234 | 34.143.73.2 | TCP | 54 | 54185 → 443 [ACK] Seq=1 Ack=1 Win=65535 Len=0 |
| 3994 | 2025-09-23 18:30:05.497675 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 571 | Client Hello (SNI=mcp-remote-822069824080.us-central1.run.app) |
| 3995 | 2025-09-23 18:30:05.533459 | 34.143.73.2 | 192.168.5.234 | TCP | 54 | 443 → 54185 [ACK] Seq=1 Ack=518 Win=65535 Len=0 |
| 3996 | 2025-09-23 18:30:05.546306 | 34.143.73.2 | 192.168.5.234 | TLSv1.3 | 2878 | Server Hello, Change Cipher Spec |
| 3997 | 2025-09-23 18:30:05.546383 | 192.168.5.234 | 34.143.73.2 | TCP | 54 | 54185 → 443 [ACK] Seq=518 Ack=2825 Win=65535 Len=0 |
| 3998 | 2025-09-23 18:30:05.546781 | 34.143.73.2 | 192.168.5.234 | TCP | 2878 | 443 → 54185 [PSH, ACK] Seq=2825 Ack=518 Win=65535 Len=2824 [TCP PDU reassemb... |
| 3999 | 2025-09-23 18:30:05.546781 | 34.143.73.2 | 192.168.5.234 | TLSv1.3 | 1207 | Application Data |
| 4000 | 2025-09-23 18:30:05.546836 | 192.168.5.234 | 34.143.73.2 | TCP | 54 | 54185 → 443 [ACK] Seq=518 Ack=6802 Win=65535 Len=0 |
| 4001 | 2025-09-23 18:30:05.549232 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 134 | Change Cipher Spec, Application Data |
| 4002 | 2025-09-23 18:30:05.549595 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 319 | Application Data |
| 4003 | 2025-09-23 18:30:05.549684 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 180 | Application Data |
| 4004 | 2025-09-23 18:30:05.584733 | 34.143.73.2 | 192.168.5.234 | TCP | 54 | 443 → 54185 [ACK] Seq=6802 Ack=989 Win=65535 Len=0 |
| 4005 | 2025-09-23 18:30:05.628689 | 34.143.73.2 | 192.168.5.234 | TLSv1.3 | 991 | Application Data, Application Data |
| 4006 | 2025-09-23 18:30:05.629650 | 192.168.5.234 | 34.143.73.2 | TCP | 54 | 54185 → 443 [FIN, ACK] Seq=989 Ack=7739 Win=64598 Len=0 |
| 4007 | 2025-09-23 18:30:05.631692 | 192.168.5.234 | 192.178.50.74 | TLSv1.3 | 480 | Application Data |

- **SYN**
 - Paquete **3990**:
 - Source: 192.168.5.234 → Destination: 34.143.73.2 (Cloud Run).
 - Info: 54185 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
- **SYN, ACK**
 - Paquete **3992**:
 - Source: 34.143.73.2 (Cloud Run) → Destination: 192.168.5.234
 - Info: 443 → 54185 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 .
- **ACK**
 - Paquete **3993**:
 - Source: 192.168.5.234 → Destination: 34.143.73.2 .
 - Info: 54185 → 443 [ACK] Seq=1 Ack=1 Win=65535 Len=0 .

En la captura de Wireshark se observa lo siguiente:

1. **SYN (Synchronize)**
 - Paquete No. **3990**
 - Origen: 192.168.5.234 (cliente) → Destino: 34.143.73.2 (servidor Cloud Run)
 - Descripción: El cliente envía un segmento TCP con el flag **SYN=1**, solicitando iniciar la conexión.
2. **SYN, ACK (Synchronize + Acknowledge)**
 - Paquete No. **3992**
 - Origen: 34.143.73.2 (servidor) → Destino: 192.168.5.234 (cliente)
 - Descripción: El servidor responde con **SYN=1, ACK=1**, indicando que acepta la conexión y confirma la recepción del SYN del cliente.
3. **ACK (Acknowledge)**
 - Paquete No. **3993**
 - Origen: 192.168.5.234 (cliente) → Destino: 34.143.73.2 (servidor)
 - Descripción: El cliente responde con un **ACK=1**, confirmando la respuesta del servidor.

Negociación TLS

Después de completar el establecimiento de la conexión TCP, el cliente y el servidor inician el proceso de **negociación TLS (Transport Layer Security)** con el fin de cifrar la comunicación.

Este proceso asegura que los mensajes JSON-RPC viajen de forma confidencial e íntegra entre el cliente MCP y el servidor remoto en Google Cloud Run.

The image displays a Wireshark network capture of a TLS handshake. The top pane shows the 'Transport Layer Security' tree with expanded fields for the Client Hello message. The middle pane shows a packet list with details for each packet. The bottom pane shows the packet bytes with a hex-to-ASCII view of the TLS segment data.

Transport Layer Security

- TLSv1.3 Record Layer: Handshake Protocol: Client Hello**
 - Content Type: Handshake (22)
 - Version: TLS 1.0 (0x0301)
 - Length: 512
- Handshake Protocol: Client Hello**
 - Handshake Type: Client Hello (1)
 - Length: 508
 - Version: TLS 1.2 (0x0303)**
 - [Expert Info (Chat/Deprecated): This legacy_version field MUST be ignored. The supported_versions extension is present and MUST be used instead.]
 - [Severity level: Chat]
 - [Group: Deprecated]
 - Random: c7f9e98e510db39fc709bc70ccdd24fdd1fd027a5804f678cf1785e363f1d796
 - Session ID Length: 32
 - Session ID: 734113a9c10124beb19b9cd74d0a85171d49a2d2fa992eac4f834d5881d0baac
 - Cipher Suites Length: 36
 - Cipher Suites (18 suites)**
 - Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
 - Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
 - Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
 - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
 - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
 - Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xc0ca9)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xc0ca8)
 - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
 - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
 - Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x009f)
 - Cipher Suite: TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x009e)
 - Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x006b)
 - Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x0067)
 - Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
 - Compression Methods Length: 1
 - Compression Methods (1 method)**
 - Compression Method: null (0)
 - Extensions Length: 399
 - Extension: server_name (len=48) name=mcp-remote-822069824080.us-central1.run.app**

tcp

| Time | Source | Destination | Protocol | Length | Info |
|------|----------------------------|-----------------|-----------------|---------|--|
| 1973 | 2025-09-23 18:30:05.282040 | 34.143.73.2 | 192.168.5.234 | TLSv1.3 | 1207 Application Data |
| 1974 | 2025-09-23 18:30:05.282040 | 200.119.162.... | 192.168.5.234 | TCP | 992 [TCP Retransmission] 443 → 54150 |
| 1975 | 2025-09-23 18:30:05.282040 | 34.143.73.2 | 192.168.5.234 | TCP | 1207 [TCP Retransmission] 443 → 54184 |
| 1976 | 2025-09-23 18:30:05.282040 | 57.144.197.33 | 192.168.5.234 | TCP | 103 [TCP Retransmission] 80 → 54181 |
| 1977 | 2025-09-23 18:30:05.282232 | 192.168.5.234 | 200.119.162.... | TCP | 66 54150 → 443 [ACK] Seq=1219 Ack=59 |
| 1978 | 2025-09-23 18:30:05.282326 | 192.168.5.234 | 34.143.73.2 | TCP | 66 54184 → 443 [ACK] Seq=518 Ack=680 |
| 1979 | 2025-09-23 18:30:05.282363 | 192.168.5.234 | 57.144.197.33 | TCP | 66 54181 → 80 [ACK] Seq=1015 Ack=205 |
| 1980 | 2025-09-23 18:30:05.285825 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 134 Change Cipher Spec, Application Data |
| 1981 | 2025-09-23 18:30:05.286274 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 318 Application Data |
| 1982 | 2025-09-23 18:30:05.286361 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 141 Application Data |
| 1983 | 2025-09-23 18:30:05.324335 | 200.119.162.... | 192.168.5.234 | TCP | 284 [TCP Spurious Retransmission] 443 |
| 1984 | 2025-09-23 18:30:05.324383 | 192.168.5.234 | 200.119.162.... | TCP | 66 [TCP Dup ACK 3977#1] 54150 → 443 |
| 1985 | 2025-09-23 18:30:05.380183 | 34.143.73.2 | 192.168.5.234 | TCP | 54 443 → 54184 [ACK] Seq=6802 Ack=94 |
| 1986 | 2025-09-23 18:30:05.380183 | 57.144.197.33 | 192.168.5.234 | TCP | 66 [TCP Dup ACK 3964#1] 80 → 54181 |
| 1987 | 2025-09-23 18:30:05.441514 | 34.143.73.2 | 192.168.5.234 | TLSv1.3 | 974 Application Data, Application Data |
| 1988 | 2025-09-23 18:30:05.443562 | 192.168.5.234 | 34.143.73.2 | TCP | 54 54184 → 443 [FIN, ACK] Seq=949 Ac |
| 1989 | 2025-09-23 18:30:05.462262 | 192.168.5.234 | 34.143.73.2 | TCP | 62 54185 → 443 [SYN] Seq=0 Win=65535 |
| 1990 | 2025-09-23 18:30:05.478554 | 34.143.73.2 | 192.168.5.234 | TCP | 54 443 → 54184 [FIN, ACK] Seq=7722 A |
| 1991 | 2025-09-23 18:30:05.478635 | 192.168.5.234 | 34.143.73.2 | TCP | 54 54184 → 443 [ACK] Seq=950 Ack=772 |
| 1992 | 2025-09-23 18:30:05.497043 | 192.168.5.234 | 34.143.73.2 | TCP | 54 443 → 54185 [ACK] Seq=1 Ack=1 Win |
| 1993 | 2025-09-23 18:30:05.497043 | 192.168.5.234 | 34.143.73.2 | TCP | 54 54185 → 443 [ACK] Seq=1 Ack=1 Win |
| 1994 | 2025-09-23 18:30:05.497675 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 571 Client Hello (SNI=mcp-remote-8220 |
| 1995 | 2025-09-23 18:30:05.533459 | 34.143.73.2 | 192.168.5.234 | TCP | 54 443 → 54185 [ACK] Seq=1 Ack=518 W |
| 1996 | 2025-09-23 18:30:05.546306 | 34.143.73.2 | 192.168.5.234 | TLSv1.3 | 2878 Server Hello, Change Cipher Spec |
| 1997 | 2025-09-23 18:30:05.546383 | 192.168.5.234 | 34.143.73.2 | TCP | 54 54185 → 443 [ACK] Seq=518 Ack=282 |
| 1998 | 2025-09-23 18:30:05.546781 | 34.143.73.2 | 192.168.5.234 | TCP | 2878 443 → 54185 [PSH, ACK] Seq=2825 A |
| 1999 | 2025-09-23 18:30:05.546781 | 34.143.73.2 | 192.168.5.234 | TLSv1.3 | 1207 Application Data |
| 4000 | 2025-09-23 18:30:05.546836 | 192.168.5.234 | 34.143.73.2 | TCP | 54 54185 → 443 [ACK] Seq=518 Ack=680 |
| 4001 | 2025-09-23 18:30:05.549232 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 134 Change Cipher Spec, Application Data |
| 4002 | 2025-09-23 18:30:05.549595 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 319 Application Data |
| 4003 | 2025-09-23 18:30:05.549584 | 192.168.5.234 | 34.143.73.2 | TLSv1.3 | 180 Application Data |
| 4004 | 2025-09-23 18:30:05.584793 | 34.143.73.2 | 192.168.5.234 | TCP | 54 443 → 54185 [ACK] Seq=6802 Ack=98 |
| 4005 | 2025-09-23 18:30:05.626869 | 34.143.73.2 | 192.168.5.234 | TLSv1.3 | 991 Application Data, Application Data |
| 4006 | 2025-09-23 18:30:05.626950 | 192.168.5.234 | 34.143.73.2 | TCP | 54 54185 → 443 [FIN, ACK] Seq=989 Ac |
| 4007 | 2025-09-23 18:30:05.631692 | 192.168.5.234 | 192.178.50.74 | TLSv1.3 | 480 Application Data |
| 4008 | 2025-09-23 18:30:05.631803 | 192.168.5.234 | 192.178.50.74 | TLSv1.3 | 842 Application Data |
| 4009 | 2025-09-23 18:30:05.680330 | 192.178.50.74 | 192.168.5.234 | TCP | 54 443 → 54180 [ACK] Seq=6274 Ack=74 |
| 4010 | 2025-09-23 18:30:05.680330 | 34.143.73.2 | 192.168.5.234 | TCP | 54 443 → 54185 [FIN, ACK] Seq=7739 A |
| 4011 | 2025-09-23 18:30:05.680416 | 192.168.5.234 | 34.143.73.2 | TCP | 54 54185 → 443 [ACK] Seq=990 Ack=774 |

Bytes sent since last PSH flag: 2824

TCP payload (2824 bytes)

[Reassembled MDU in frame: 3999]

TCP segment data (2691 bytes)

Transport Layer Security

- TLSv1.3 Record Layer: Handshake Protocol: Server Hello**
 - Content Type: Handshake (22)
 - Version: TLS 1.2 (0x0303)
 - Length: 122
- Handshake Protocol: Server Hello**
 - Handshake Type: Server Hello (2)
 - Length: 118
 - Version: TLS 1.2 (0x0303)**
 - [Expert Info (Chat/Deprecated): This legacy_version field MUST be ignored. The supported_versions extension is present and MUST be used instead.]
 - [Severity level: Chat]
 - [Group: Deprecated]
 - Random: fcb36af9be8d24d044c6b5b1ef7183db3a7adb58125a49e6b95fea8e38e3
 - Session ID Length: 32
 - Session ID: 734113a9c10124beb19b9cd74d0a85171d49a2d2fa992eac4f834d5881d
 - Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
 - Compression Method: null (0)
 - Extensions Length: 46
 - Extensions: key_share (len=36) x25519**
 - Type: key_share (51)
 - Length: 36
 - Key Share extension**
 - Key Share Entry: Group: x25519, Key Exchange length: 32
 - Extension: supported_versions (len=2) TLS 1.3**
 - Type: supported_versions (43)
 - Length: 2
 - Supported Version: TLS 1.3 (0x0304)
 - [JA3S Fullstring: 771,4866,51-43]
 - [JA3S: 907bf3ecf1c987c89946b737b43de8]
 - TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec**
 - Content Type: Change Cipher Spec (20)
 - Version: TLS 1.2 (0x0303)
 - Length: 1
 - Change Cipher Spec Message
 - TLS segment data (2691 bytes)

1. Client Hello

- El cliente anuncia las versiones de TLS que soporta (TLS 1.3 y 1.2).
- Propone múltiples *cipher suites* (TLS_AES_256_GCM_SHA384 , TLS_CHACHA20_POLY1305_SHA256 , entre otros).
- Incluye la extensión **SNI (Server Name Indication)** con el nombre del servidor remoto:
mcp-remote-822069824080.us-central1.run.app .

2. Server Hello

- El servidor responde seleccionando la versión **TLS 1.3**.

- Elige el *cipher suite* `TLS_AES_256_GCM_SHA384` como mecanismo de cifrado.
- Se incluye el intercambio de claves basado en curvas elípticas (`x25519`).
- Posteriormente aparecen los mensajes **Change Cipher Spec**, indicando que todo el tráfico posterior viajará cifrado.

Requests JSON-RPC

Con el canal TLS ya establecido, el cliente MCP comienza a enviar solicitudes en formato **JSON-RPC** hacia el servidor remoto desplegado en Google Cloud Run.

En Wireshark, estos mensajes aparecen como **TLSv1.3 Application Data**, ya que el contenido JSON está cifrado y no es visible directamente. No obstante, a partir de las pruebas realizadas en el cliente se sabe que las solicitudes transmitidas corresponden a:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "initialize",
  "params": {}
}
```

Esta petición corresponde al primer mensaje que establece la sesión con el servidor remoto.

```
{
  "jsonrpc": "2.0",
  "id": 2,
  "method": "tools/call",
  "params": { "name": "current_time" }
}
```

Esta petición instruye al servidor remoto a ejecutar la herramienta definida (`current_time`), que devuelve la hora UTC actual.

Responses JSON-RPC

Una vez recibidas las solicitudes JSON-RPC, el servidor remoto responde con los resultados correspondientes.

Respuesta a la inicialización

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": { "status": "ok" }
}
```

Respuesta a la llamada de herramienta (`tools/call`)

```
{
  "jsonrpc": "2.0",
  "id": 2,
  "result": { "result": "2025-09-23 15:45:56 UTC" }
}
```

El servidor ejecuta la herramienta solicitada (`current_time`) y devuelve como resultado la hora UTC actual.

Observaciones a nivel de capas

Capa de Enlace de Datos

- Se observa la transmisión de tramas **Ethernet II**.
- Cada trama contiene las direcciones **MAC origen y destino**:
 - Origen: interfaz de red del cliente local.
 - Destino: gateway/router (MAC de la puerta de enlace).
- El protocolo de nivel superior encapsulado es IPv4.
- Esta capa se encarga de la **entrega punto a punto** dentro de la red local.

Capa de Red

- El protocolo utilizado es **IPv4**.
- Direcciones identificadas:
 - Cliente: 192.168.5.234 (IP privada local).
 - Servidor: 34.143.73.2 (IP pública de Google Cloud Run).
- Se observa el enrutamiento de paquetes desde la red privada hasta Internet.
- Esta capa gestiona la **dirección lógica y el encaminamiento** de los paquetes.

Capa de Transporte

- El protocolo es **TCP**.
- Establecimiento de conexión mediante el **3-way handshake**:
 - SYN → SYN/ACK → ACK.
- El puerto destino es el **443 (HTTPS)**.
- Una vez establecida la conexión TCP, se negocia una sesión **TLSv1.3**, que asegura la confidencialidad e integridad de los datos.
- TCP proporciona transmisión **fiable y orientada a conexión**.

Capa de Aplicación

- El tráfico de aplicación corresponde al protocolo **HTTPS**, dentro del cual viajan mensajes **JSON-RPC 2.0**.
- Solicitudes detectadas:
 - `initialize` (sincronización de la sesión).
 - `tools/call` con parámetro `current_time`.
- Respuestas del servidor:
 - Confirmación de inicialización (`status=ok`).
 - Resultado de la herramienta (hora UTC).
- Aunque en Wireshark se ve como **TLS Application Data** (cifrado), mediante pruebas directas se confirma el contenido JSON.
- Esta capa se encarga de la **lógica de negocio y la comunicación entre cliente-servidor**.

Conclusiones y opinión

El desarrollo e implementación de servidores MCP permitió comprender a cómo se estructura y se lleva a cabo la comunicación basada en el protocolo JSON-RPC, tanto en un entorno local como remoto. Se trabajó con diferentes servicios en la nube, en particular **Google Cloud Run**, lo que aportó experiencia en despliegue de aplicaciones en entornos gestionados y en la integración de servicios distribuidos.

A nivel técnico, se observaron las distintas capas de comunicación (enlace, red, transporte y aplicación), comprobando el proceso completo desde la **sincronización TCP**, la **negociación TLS**, hasta el intercambio de **mensajes JSON-RPC**. Asimismo, la captura y análisis con **Wireshark** permitió visualizar el tráfico y relacionar la teoría con la práctica de forma tangible.

En cuanto a la integración de servidores desarrollados por compañeros (Filesystem, Spotify, Movies, League of Legends), se evidenció la versatilidad del estándar MCP para soportar escenarios diversos, desde la manipulación de archivos hasta la recomendación de películas o configuraciones en videojuegos. Esto demuestra el potencial de MCP para interoperar con múltiples servicios y aplicaciones en un mismo ecosistema.

Desde el punto de vista personal, el proyecto resultó **relevante y actual**, al estar alineado con las tendencias de protocolos abiertos e integración modular de servicios. Sin embargo, la experiencia también mostró algunas dificultades: las **instrucciones en ciertos momentos no son del todo intuitivas**, lo que demandó mayor tiempo de interpretación y prueba. Además, la **implementación de los servidores de los compañeros se volvió complicada**, debido a que cada persona organizó y documentó su proyecto de forma distinta, lo que generó inconsistencias y dificultó la integración final.

En síntesis, el proyecto fue valioso tanto por el aprendizaje técnico como por la experiencia de trabajo colaborativo, aunque se resalta la importancia de mejorar la **estandarización de instrucciones y documentación** para que la integración entre equipos sea más fluida.