Lic. Gabriel Chaldu
Laboratorio IV

HTTP en una API REST

Una **petición HTTP** en el contexto de una **API REST** (Representational State Transfer) es una solicitud que un cliente (como un navegador, una aplicación móvil o un sistema) realiza a un servidor para interactuar con recursos (datos o servicios) a través de un protocolo estándar, como HTTP.

En REST, cada recurso (p. ej., un usuario, un producto) se representa mediante una **URI** (Uniform Resource Identifier), y las operaciones que se realizan sobre estos recursos se corresponden con los diferentes **métodos HTTP**. Los métodos más comunes son:

Tipos de peticiones HTTP en una API REST:

1. **GET**:
    - **Función**: Recuperar datos o recursos del servidor.
    - **Ejemplo**: Obtener una lista de usuarios o los detalles de un usuario específico.
    - **Ejemplo de URI**: GET /usuarios (todos los usuarios), GET /usuarios/{id} (un usuario específico).
2. **POST**:
    - **Función**: Crear un nuevo recurso en el servidor.
    - **Ejemplo**: Crear un nuevo usuario enviando los datos necesarios en el cuerpo de la solicitud.
    - **Ejemplo de URI**: POST /usuarios.
3. **PUT**:
    - **Función**: Actualizar completamente un recurso existente. Se envía el recurso completo con las modificaciones.
    - **Ejemplo**: Actualizar todos los detalles de un usuario específico.
    - **Ejemplo de URI**: PUT /usuarios/{id}.
4. **PATCH**:
    - **Función**: Actualizar parcialmente un recurso existente. Se envían solo los campos que deben ser modificados.
    - **Ejemplo**: Cambiar solo el correo electrónico de un usuario.

    - **Ejemplo de URI**: PATCH /usuarios/{id}.

5. **DELETE**:
    - **Función**: Eliminar un recurso del servidor.
    - **Ejemplo**: Eliminar un usuario específico.
    - **Ejemplo de URI**: DELETE /usuarios/{id}.
6. **OPTIONS**:
    - **Función**: Obtener información sobre las capacidades del servidor en cuanto a las operaciones permitidas sobre un recurso.
    - **Ejemplo**: Ver qué métodos HTTP están permitidos para un recurso.

Resumen del flujo:

- **Cliente** → Envía la petición HTTP con un método específico (GET, POST, PUT, etc.).
- **Servidor** → Recibe la petición, procesa la solicitud y devuelve una respuesta, que puede incluir datos (normalmente en formato JSON) o un código de estado HTTP (p. ej., 200 OK, 404 Not Found, 500 Internal Server Error, etc.).

Este modelo de peticiones es fundamental para la arquitectura REST, que busca ser escalable y sencilla, usando operaciones estándar de HTTP para manipular recursos.