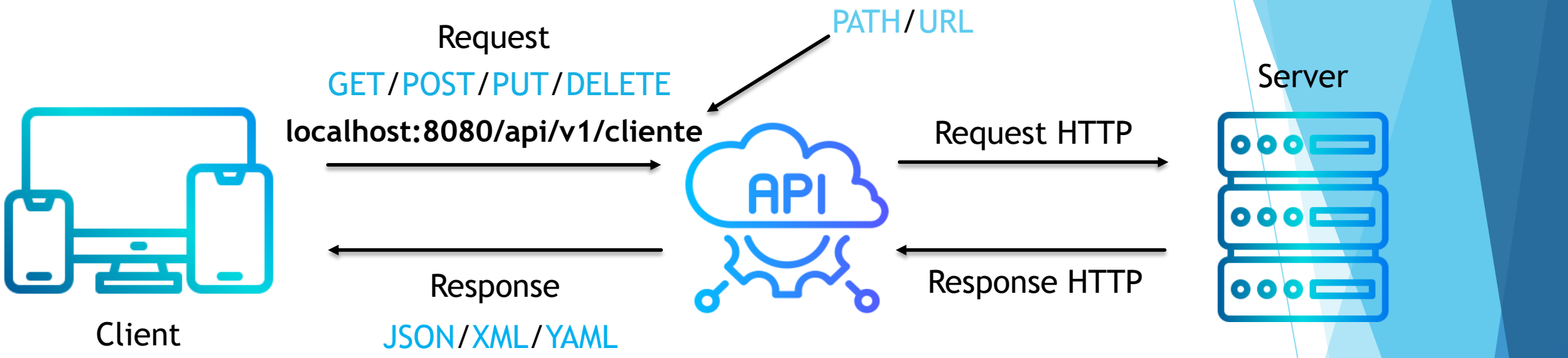
¿Que es un Api

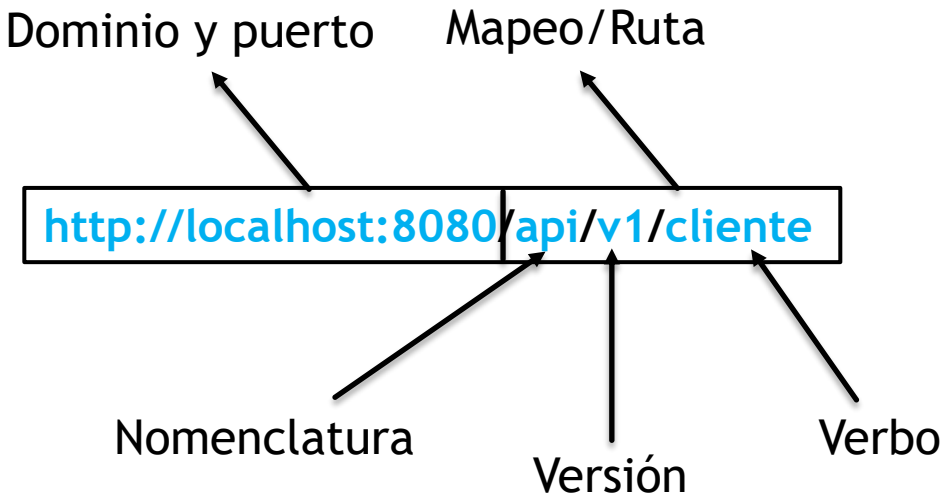
1. Application Programming Interface (Interfaz de programación de aplicaciones.
2. Conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

¿Que es Rest?

- Representational State Transfer:
- Interfaz para conectar varios sistemas basados en el protocolo HTTP (uno de los protocolos más antiguos) y sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON.
- Rest se apoya en los verbos HTTP: **GET, POST, PUT, DELETE** entre otros:
  - **Post:** Crear recursos o registros nuevos.
  - **Get:** Obtener un listado o recurso en concreto.
  - **Put:** Modificar un recurso completo o si no existe crea uno nuevo.
  - **Patch:** Modificar parcialmente un recurso.
  - **Delete:** Borrar un registro o recurso.

Como funciona un Api Rest.





Características de Rest

- **Protocolo cliente/servidor sin estado:** Cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla.
- **Los objetos en REST siempre se manipulan a partir de la URI.**
- **Interfaz Uniforme:** Para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI.
- **Sistema de capas:** Arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.

Protocolo HTTP

Hypertext Transfer Protocol
Permite conectarse a un recurso o servicio.
Permite las solicitudes(**Requests**) y respuestas(**Responses**) entre Cliente-Servidor.

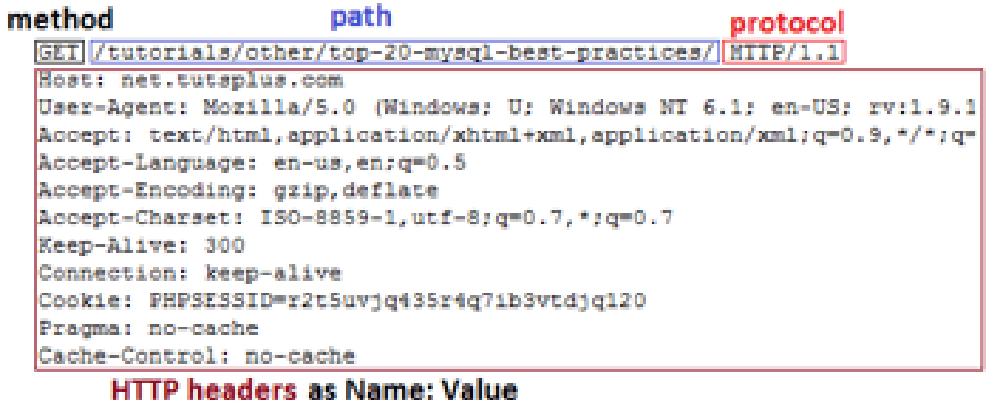
GET	Permite obtener recursos. CRUD: SELECT
POST	Inserta un nuevo recurso. CRUD: CREATE - INSERT
PUT	Actualiza el recurso completo. CRUD: UPDATE
PATCH	Actualiza el recurso parcialmente. CRUD: SELECT
DELETE	Borra un recurso existente. CRUD: SELECT

HTTP REQUEST

Permite enviar todo tipo de petición HTTP a un URL especifico y procesar la respuesta del servidor HTTP.

Estructura de un HTTP request

- **Method:** GET, POST, PUT, PATCH, DELETE, HEAD, etc. Indica que tipo de petición es.
- **Path:** La url que se solicita, donde se encuentra el recurso.
- **Protocolo:** Contiene HTTP y su versión , actualmente 1.1.
- **Headers:** Son esquemas de **key: value** que contienen información sobre el HTTP request y el navegador.
- **Body:** Si se envía información al servidor a través del POST o PUT, esta va en el body.



HTTP Response

Es el mensaje que envía el servidor al cliente tras haber recibido una petición HTTP request.

HTTP response – Status Code

Códigos de éxito

- **200 OK :** La solicitud se ha realizado correctamente (GET – POST - PUT).
- **201 Created:** Creación de un nuevo recurso (Devuelve el mensaje en el body, POST - PUT).
- **202 Accepted:** La solicitud se ha aceptado para su procesamiento y el procesamiento no se ha completado ni iniciado(Lotes).
- **204 No Content:** La solicitud se ha realizado correctamente, pero no es necesario que devuelva un cuerpo.

HTTP Response

Es el mensaje que envía el servidor al cliente tras haber recibido una petición HTTP request.

HTTP response – Status Code

Códigos de error de cliente

- **400 Bad Request:** El servidor no puede procesar la solicitud.
- **401 Unauthorized:** La solicitud no se ha aplicado por que carece de credenciales de autenticación validas para el recurso de destino.
- **403 Forbidden:** El servidor entendió la solicitud pero se niega a autorizarla.
- **404 Not Found:** El servidor no puede encontrar nada que coincida con la URI de la solicitud.
- **405 Method Not Allowed:** El servidor conoce el método de la solicitud, pero el recurso de destino no lo admite.

HTTP Response

Es el mensaje que envía el servidor al cliente tras haber recibido una petición HTTP request.

HTTP response – Status Code

Códigos de error de cliente

- **500 Internal Server:** El servidor encontró una condición inesperada que le impidió cumplir con la solicitud.
- **503 Service Unavailable:** El servidor no esta disponible actualmente y no esta listo para manejar ninguna solicitud debido a una sobrecarga temporal o mantenimiento.

Errores mas comunes

Estructura de recursos
Asigna correctamente los nombres de tus URL en base a cada método HTTPS

ERROR

- **GET:**
  - /api/listarClientes
  - /api/cliente-eliminar/14
  - /api/cliente/guardar/25
- **POST:**
  - /api/listarClientes
  - /api/cliente/eliminar/24
  - /api/cliente/actualizar?id25
- **PUT:**
  - /api/cliente/actualizar?id=25&nomnbre=Pablo&apellido=Mora&edad=32

CORRECTO

- **GET:**
  - /api/v1/clientes
  - /api/v1/cliente?id=12
  - /pi/v1/cliente/2/telefonos
  - /api/v1/clientes?page=2&size=10&order=desc
- **POST:**
  - /api/v1/cliente
  - /api/v1/usuario/12/restablecer-clave
- **PUT:**
  - /api/v1/cliente
  - /api/v1/cliente/1
  - /api/v1/cliente?id=12
- **Delete:**
  - /api/v1/cliente?id=12

Especifica los status code

Define los status code según su método.

Warning

- **GET:**
  - 200 OK
- **POST:**
  - 200 OK
- **PUT:**
  - 200 OK
  - 210 CREATE

Correcto

- **GET - DELETE:**
  - 200 OK
- **POST:**
  - 200 OK
  - 201 CREATE
- **PUT:**
  - 200 OK
  - 210 CREATE
- **Error:**
  - 404 NOT FOUND
  - 405 METHOD NOT ALLOWED

Estructura del body response

Debe ser claro.

Warning

- **POST:**
  - Guardado correctamente.
- **PUT:**
  - Modificado correctamente.
- **DELETE:**
  - Eliminado correctamente.

Correcto

- **PUT - POST - DELETE:**

  {

  codigo=0,

  mensaje="Modificado/Guardado correctamente,

  error=false,

  fecha=2023/08/04

  }

- **PUT (Excepciones):**

  {

  codigo=2,

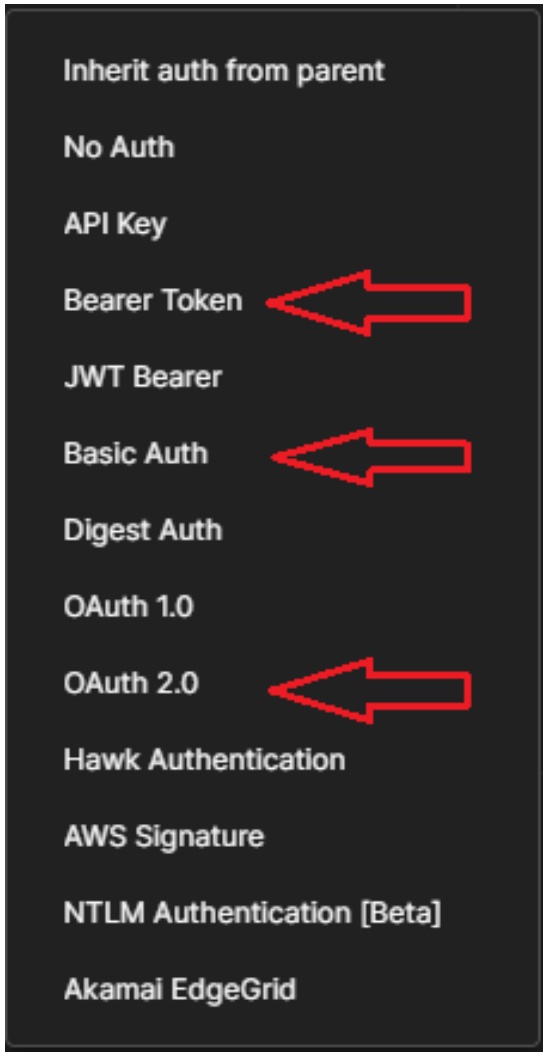
  mensaje=1,

  error=false,

  fecha=2023/08/04

  }

Seguridad en de Api.



- **Bearer Token:**
  - Es un esquema de autenticación HTTP en el cual la seguridad se apoya en el uso de cadenas de texto encriptadas, normalmente generadas por el servidor y que identifican al portador(**Bearer**) del mensaje mediante la inclusión de dichas cadenas (**Token**) en todas la peticiones de recursos realizadas al servidor.
- **Basic Auth:**
  - Mecanismo de autenticación mas sencillo admitido por HTTP e implica que el cliente envía el nombre de usuario y la contraseña como texto codificado en base64 sin cifrar.
- **OAuth 2.0:**
  - Método de autorización utilizado por compañías como Google, Facebook, Twitter, Amazon, Microsoft, etc. Su propósito es permitir a otros proveedores, servicios o aplicaciones el acceso a la información sin facilitar directamente las credenciales de los usuarios.

Documentación del Api con Swagger

- **Swagger:**
  - Serie de reglas, especificaciones y herramientas que ayudan a documentar las APIs, de esta manera se logra que la documentación sea realmente útil para las personas que las necesitan.