Informe API REST

1. Introduccion.

1. API ¿Qué es ?

Una Api (Interfaz de programacion de aplicaciones) es un conjunto de reglas, protocolos y heramientas que permiten la comunicación entre diferentes sistemas de software.Actúa como una puerta de enlace entre los clientes y los recursos.

I. Funcionamiento de las APIs

Las APIs permiten que las aplicaciones interactuen mediante solicitudes y respuestas.Un sistema expone una API a través de una interfaz, permitiendo que otros sistemas accedan a datos y funciones sin conocer la complejidad intena del servicio.Las organizaciones utilizan APIs para compartir recursos de manera segura, controlando que clientes tienen acceso a determinados datos.

II. Tipos de APIs

Existen diferentes tipos de API según el metodod de comunicación:

- **REST (Representational state transfer):** Utiliza HTTP para solicitar y manipular recursos de manera sencilla y escalable.
- **SOAP (Simple Object Access Protocol):** Un protocolo más estructurado que emplea XML para el intercambio de datos.
- **RPC(Remote Procedure Call):** Permite invocar funciones en sistemas remotos a traves de la red.
- **WebSocket:** Proporciona una comunicación bidireccional en tiempo real entre cliente y servidor.

2. API REST ¿Qué es?

Una APIREST es una interfaz de programacion de aplicaciones (API) que permite la comunicación entre diferentes sistemas informaticos a través de Internet utilizando el protocolo HTTP.Se basa en los principios del estilo arquitectonico REST (Representational State Transfer), definido en el baño 2000 por el Dr.Roy Fielding, lo que garantiza una estructura ligera, flexible, escalable y eficiente.

I. Como funcionan las API REST

Las API REST utilizan metodos estandar de HTTP para realizar operaciones CRUD (Crear, Leer, Actualizar y Borrar) en los recursos.Por ejemplo:

- **GET** Recupera un recurso.
- **POST** Crea un nuevo recurso.
- **PUT** Actualiza un recurso existente.
- **DELETE** Elimina un recurso.

Cada recurso tiene una representacion en formatos como **JSON, XML o texto plano, siendo JSON** el mas popular por su legibilidad y compatibilidad con multiples lenguajes de programación.

Las llamadas a una API REST tambien incluyen encabezados y parametros que contienen informacion crucial, como identificadores de recursos (URI), metadatos, autenticaciones y almacenamineto en caché.Además, cada solicitud HTTP es independinete (stateless), lo que significa que no se almacena información sobre sesiones entre el cliente y el servidor.

II. Mejores prácticas en el diseño de API Rest

Para garantizar seguridad y eficiencia, una API REST debe seguir ciertos principios:

- **OpenAPI Specification (OAS3):**Permite documentar la API de forma estandarizada para que desarrolladores comprenden sus funcionalidades y parámetros.
- **Autenticacion y Seguridad:** Se recomienda el uso de HTTPS para una transmision segura,junto con metodos como Oauth 2.0 para restringir accesos no autorizados.
- **Validacion de parámetros y tokens web JSON(JWT):**Asegura que solo clientes autorizados accedan a la API.
- **Usos de HATEOAS(Hypermedia as the Engine of Application State)** para mejorar la navegacion entre recursos.

III. Aplicaciones y Ventajas.

Las API REST son fundamentales en arquitecturas de microservicios, permitiendo la conexión fluida entre distintas aplicaciones.Se usan en diversos sectores,desde servicios bancarios hasta plataforma de redes sociales.Su simplicidad, adaptabilidad y compatibilidad con multiples lenguajes de programacion la convierten en una opcion ideal para desarrolladores y empresas que buscan servicios web escalables y eficientes.

En resumen, una **API REST es un puente entre aplicaciones**, asegurando una comunicación ordenada y eficaz con estandares bien definidos.

APIS MAS POPULARES EN EL MUNDO

3. REST

La transferencia de estado representacional es una arquitectura de software que establece condiciones sobre el funcionamiento de una API.Inicialmente, se diseño para gestionar la comunicación en redes complejas, como internet, permitiendo conexiones eficientes y faciles de modificar.

I. Conceptos clave de REST

Las API REST son aquellas que siguen principios de REST, mientras que los servicios web RESTful implementan esta arquitectura.

II. Principios de REST

- **1.Interfaz uniforme**: El servidor transfiere recursos en un formato estándar (JSON, XML, HTML), permitiendo su identificación mediante **URI**. Además, los clientes reciben metadatos y enlaces para descubrir nuevos recursos dinámicamente.
- **2.Tecnología sin estado** (*stateless*): Cada solicitud es independiente, por lo que el servidor no almacena información de sesiones previas.
- **3.Sistema por capas**: Las aplicaciones pueden tener múltiples niveles (seguridad, lógica empresarial) que trabajan de manera conjunta sin que el cliente perciba estas capas
- **4.Almacenamiento en caché**: Permite mejorar la velocidad de respuesta guardando ciertos datos temporalmente en la memoria del cliente o intermediarios.
- **5.Código bajo demanda** (*opcional*): Los servidores pueden enviar código al cliente para mejorar su funcionalidad, como la validación de formularios en tiempo real.

2. Principios Clave.

4. Términos básicos de API Rest

- **Endpoint**: URL que especifica donde se accede a un recurso, como { ***/productos*** } en una tienda online.
- **Método HTTP**: Define la accion sobre el recurso (GET, POST, PUT, DELETE).
- **URI**: Identificador de recurso, compuesto por ***endpoint+ parámetros.***
- **Parámetros:** Informacion extra en la solicitud, que puede ir en la URL o el cuerpo de la peticion.

- **Payload**: Datos adicionales enviados con la solicitud, generalmente en JSON o XML.
- **Autenticación**: Verifica la identidad del usuario mediante credenciales en la solicitud HTTP.
- **Respuesta HTTP :** Respuesta del servidor , con datos o mensajes de error.
- **Códigos de estado HTTP**: Indican el resultado de la solicitud.(**200 OK, 400 Bad Request, 401 Unauthorized, 404 Not Found, 500 Internal Server Error**).
- **Cache:**Almacenamiento temporal de respuestas para mejorar rendimiento y reducir craga en el servidor.
- **API Documentation**: Conjunto de guías y especificaciones que describen cómo interactuar con una API. Incluye métodos, endpoints y ejemplos de uso.
- **API Portal:** Plataforma que permite a desarrolladores acceder, gestionar y probar APIs. Facilita la integración y documentación.
- **API Endpoint:** Dirección URL específica donde un cliente puede acceder a un recurso dentro de la API.
- **API Call**: Petición realizada por el cliente a un endpoint de la API para obtener o manipular datos.
- **API Layer**: Capa dentro de una arquitectura que actúa como intermediaria entre el cliente y el servidor para gestionar solicitudes.
- **API Request**: Mensaje enviado por un cliente a un servidor, incluyendo parámetros, encabezados y datos para una operación.
- **API Lifecycle**: Proceso completo desde el diseño, implementación, prueba, mantenimiento y eventual retiro de una API.
- **API Integration**: Conexión entre una API y otras aplicaciones o sistemas para intercambiar datos y funcionalidades.
- **API Gateway**: Servicio que gestiona el tráfico de solicitudes hacia múltiples APIs, proporcionando autenticación, seguridad y control de acceso.
- **API Keys**: Credenciales únicas utilizadas para autenticar y autorizar el acceso de un cliente a una API.

5. ¿Qué contiene la solicitud del cliente de la API RESTful?

- **Identificador unico de recursos**: Cada recurso tiene una url que especifica su ubicacione n el servidor.
- **Metodo HTTP** : Indica la accion a realizar sobre el recurso.
  - GET
  - POST

- PUT
- DELETE

- **Encabezados HTTP** ; Contienen metadatos sobre la solicitud, como formato de datos y estado de la conexión.
- **Parametros**:Detalles adicionales en la solicitud
  - Parametros de ruta(incluyen información dentro de la URL).
  - Parametros de consulta(filtran o modifican los datos).
  - Parametros de cookie(autenticación de clientes).

6. Seguridad en API REST

- **Uso de HTTPS** para encriptar la comunicación
- **Autenticación con OAuth 2.0** para gestionar accesos seguros.
- **Validacion de entradas** para prevenir ataques de inyeccion de datos.

3. Metodos HTTP.

7. GET – Recuperar información

Obtener datos de un recurso sin modificarlo.

- Idempotente: llamarlo varias veces no cambia el estado del servidor.
- Puede aceptar parametros de consulta para filtrar resultados.
- Su respuesta se puede almacenar en cache para mejorar el redimiento.

Ej; { **GET /usuarios/123** } → retorna los datos del ususario con el ID 123.

8. POST – Crear un recurso

Permite enviar datos al servidor para crear un nuevo recurso.

- Repetir la misma solicitud puede generar múltiples registros
- Se usa en formularios, registros de usuarios o inserción de datos.
- La información se envía en el cuerpo de la solicitud.

Ej; **POST /productos**

```
Content-Type: application/json

{
      "nombre": "Laptop",
      "precio": 1200
```

```
}
```

→ Crea un nuevo producto con los datos proporcionados

9. PUT – Actualizar un recurso

Sse usa para modificar completamente un recurso existente.

- Si se envia la misma solicitud varias veces, el resultado no cambia.
- Reemplaza la infromacion anterior con los nuevos datos enviados.

Ej;

PUT /usuarios/123

```
Content-Type: application/json

{
        "nombre": "Juan Pérez",
        "email": "juan@example.com"
}
```

→ Modifica la infromacion del usuario con ID 123

10. DELETE – Eliminar un recurso

Este metodo se utiliza para borrar un recurso del servidor.

- Puede requerir autenticacion para evitar eliminaciones accidentales.
- No siempre devuelve contenido en la respuesta, pero suele incluir el codigo 204 No Content si fue exitoso

Ej; { **DELETE /productos/45** } → Elimina el producto con ID 45.

4. Formato de datos.

11. JSON (JavaScript Object Notation)

El JSON es el formato mas utilizado por API REST debido a su simplicidad, legibilidad y compatibilidad con distintos lenguajes de programacion.Se basa en una estructura de pares clave-valor y utiliza corchetes {} para representar objetos y listas [] para arrays.

```json
{
  "nombre": "Juan",
  "edad": 30,
  "correo": "juan@example.com"
}
```

12.XML (Extensible Markup Language)

El XML fue am`pliamente usado en servicios webs antes que JSON se volviera estandar.Es estructurado, basado en etiquetas y permite definir reglas más estrictas sobre los datos

```xml
<usuario>
  <nombre>Juan</nombre>
  <edad>30</edad>
  <correo>juan@example.com</correo>
</usuario>
```

13.Texto Plano (Plain Text)

En algunas API REST , la respuesta o solicitud puede ser simplemente texto sin formato, ideal para transmitir mensajes simples, como notificaciones o datos de estado

```
Usuario creado exitosamente
```

14. YAML (Yet Another Markup Language)

El YAML es una alternativa a JSON que mejora la legibilidad al eliminar estructuras de llave-valor explicitas, usando identacion.

```yaml
usuario:
  nombre: Juan
  edad: 30
  correo: juan@example.com
```

15. CSV (Comma-Separated Values)

El CSV se usa en API REST para intercambiar datos tabulares, como informes financieros o listas de usuarios.

```csv
nombre, edad, correo
Juan, 30, juan@example.com
Maria, 25, maria@example.com
```

5. Ventajas y Desventajas de usar API REST.

16. Ventajas API REST

Simplicidad y facilidad de uso REST utiliza HTTP como protocolo de comunicación, lo que lo hace facil de implementar y comprender.No requiere una estructura rigida como SOAP.

Escalabilidad gracias a su arquitectura stateless, REST permite manjear grandes volumenes de solicitudes sin que el servidor almacene informacion de esatdo, lo que favorece su escalabilidad.

Flexibilidaad y compatibilidad las API REST pueden enviar y recibir datos en multiples formatos, como JSON, XML o texto plano, lo que permite su integracion con una gran variedad de aplicaciones y lenguajes.

Interoperabilidad REST permite que diferentes sistemas, independientes de la plataforma, se comuniquen sin problemas, facilitando la integracion de aplicaciones web,moviles y empresariales.

Rapidez y eficiencia las solicitudes GET pueden almacenarse en cáche, reduciendo la carga del servidor y mejorando el rendimiento de la aplicación.

Uso optimizado de HTTP aprovecha los metodos HTTP estandar (GET,PUT…) para manipular los recursos, lo que simplifica el desarrollo y la administracion de APIs.

17. Desventajas de API REST

Sin estado (stateless) cada solicitud es imdependiente, lo que significa que para operaciones complejas puede requerirse el envio repetido de datos de autenticación y contexto.

Falta de estándares estrictos REST permite mucha flexibilidad, lo que puede generar inconsistencias entre diferentes implementaciones de API, difilcultando la interoprabilidad sin una documentación clara.

Seguridad y autenticación Si no se implementa correctamente, las API REST pueden ser vulnerables a ataques, como inyección de datos, acceso no autorizado o robo de información. Se deben utilizar técnicas como OAuth 2.0 y tokens JWT para mejorar la seguridad.

No es ideal para sistemas en tiempo real REST funciona bien para operaciones estándar, pero no es la mejor opción para comunicaciones en tiempo real. Para estos casos, WebSockets o GraphQL pueden ser más adecuados.

Carga en el servidor en consultas complejas Si la API no está bien optimizada, realizar múltiples solicitudes para obtener datos relacionados puede generar una sobrecarga innecesaria en el servidor, afectando el rendimiento.

6. Recursos.

- **Postman :** Plataforma para probar API y automatizar pruebas.

- **Swagger (OPEN API) :** Generador de documentacion interactiva.
- **Apigee :** Solucion para gestionar API en la nube
- **Django REST Framework** y **Express.js**: Frameworks para crear API REST en Python y Node.js.

7. Enlaces.

MDN Web Docs

OAuth 2.0

OpenAPI Specification

POSTMAN