Junta de Andalucía

Consejería de la Presidencia, Interior, Diálogo Social y Simplificación Administrativa
Agencia Digital de Andalucía

Dirección General de Estrategia Digital y Gobierno Abierto

Financiado por la Unión Europea

OpenAPI

Normativa, especificación y directrices para el uso de OpenAPI

HOJA DE CONTROL DEL DOCUMENTO

Información del Documento			
Título	OpenAPI		
Asunto	Normativa, especificación y directrices para el uso de OpenAPI		
Nombre del fichero	OT-I_NORM_Uso de la especificación OpenAPI_v01r06.odt		
Versión	<v01r06>	**Fecha versión**	12/09/2024
		N.º Total Páginas	17

Control de Versiones			
Versión	**Descripción de los cambios**	**Elaborado por**	**Fecha Elaboración**
v01r00	Elaboración inicial del documento	OT-I	12/11/2019
V01r01	Uso de Open-API y Swagger	OT-I	04/02/2020
V01r02	Mejoras en el documento e inclusión de ejemplos	OT-I	30/04/2020
V01r03	Revisión y ajuste de formato	OT-I	20/11/20
V01r04	Inclusión de nuevos apartados	OT-I	24/11/20
V01r05	Actualización nueva plantilla	OT-I	28/01/21
V01r06	Actualización encabezamiento y pie a la nueva versión	OT-I	12/09/2024

Lista de Distribución	
Apellidos, Nombre	**Cargo / Función**

ÍNDICE

1. INTRODUCCIÓN

La Arquitectura Orientada a Servicios (SOA) es un estilo arquitectónico de TI basado en la comunicación entre servicios distribuidos débilmente acoplados y altamente interoperables.

La industria comenzó a fabricar pilas de productos SOA por fabricante cuya complejidad llegó a ser muy elevada e incompatible, en ciertas ocasiones, entre ellas.

La complejidad del uso de las pilas de productos SOA, la incompatibilidades entre ellas y la llegada de los primeros smartphones y sus limitadas capacidades de cálculo y almacenamiento, migraron del XML al JSON y de los Webservices SOAP a las comunicaciones REST.

El conjunto de todos los servicios REST que expone una aplicación u organización forman lo que denominamos API REST. Existen numerosas herramientas en el mercado para la gestión de la documentación de una API REST, donde se incluyen las guías de implementación de los servicios web con las pautas y el formato a tener en cuenta de cara a la construcción de un cliente web que acceda a dichos servicios.

Para garantizar las comunicaciones formales, completas y de calidad entre las APIs de la organización, desde la Oficina Técnica de Interoperabilidad promovemos que cuando se proceda con el desarrollo de una API REST se haga uso de la especificación OpenAPI versión 3.0 para su definición.

1.1. Alcance

Este procedimiento va dirigido a todo el personal con responsabilidades en la toma de decisiones cuando los desarrollos impliquen la creación de servicios REST.

- Directores de proyectos, que han de velar por el conocimiento y puesta en práctica de la normativa por parte de los equipos de desarrollo.
- Equipo de desarrollo, que ha de cumplir la especificación aquí presentada.
- Equipo de Interoperabilidad, responsable de la Plataforma y del gobierno de la interoperabilidad.

El **ámbito de aplicación es la Dirección General de Estrategia Digital y Gobierno Abierto de la Consejería de Presidencia, Administración Pública e Interior**, siendo una recomendación fuera de este contexto.

1.2. Condiciones de Uso

Las normas recogidas en esta guía son de **obligado cumplimiento**, dentro del alcance especificado. La Oficina Técnica de Interoperabilidad de la Consejería de Presidencia, Administración Pública e Interior (en adelante OT-I) se reserva el derecho a la modificación de la norma sin previo aviso, tras lo cual, notificará del cambio a los actores implicados para su adopción inmediata.

La OT-I podrá estudiar los casos excepcionales, en el caso de que algún actor considere necesario el incumplimiento de alguna de las normas y/o recomendaciones, deberá aportar previamente la correspondiente justificación fehacientemente documentada de la solución alternativa propuesta, así como toda aquella documentación que le sea requerida para proceder a su validación técnica.

Tras el análisis de la información aportada, la OT-I informará de manera explícita sobre las conclusiones obtenidas para lograr encontrar una solución adaptada en la medida de lo posible a las directrices marcadas.

2. NORMAS Y BUENAS PRÁCTICAS

A continuación se presenta un resumen de las normas y buenas prácticas de aplicación en el contexto del desarrollo.

Para más información, consulte los apartados que desarrollan los contenidos.

2.1. Definición de la API.

2.1.1. Adoptar una estrategia Contract-first

Se debe adoptar una estrategia de desarrollo Contract-first (Primero especificación de la API y luego desarrollos).

2.1.2. Utilizar correctamente los métodos HTTP para la definición de la API REST

Se deben utilizar correctamente los métodos HTTP en función de la acción a realizar:

Operación	Método HTTP
Listar	GET
Crear	POST
Leer	GET
Actualizar	PATCH / PUT
Borrar	DELETE

2.1.3. Utilizar última versión disponible del estandar OpenAPI

Cuando se desarrolle un nuevo servicio, se debe implementar el estandar de OpenAPI en su última versión disponible. Se puede consultar en el apartado "Specification/Latest" en la siguiente URL:

https://www.openapis.org/

2.2. Usar el lenguaje de implementación más apropiado en cada caso

El lenguaje así como la forma de implementar la API serán de libre elección según las necesidades de cada proyecto. Desde la OT-I se recomienda partir de los esqueletos que pueden generarse mediante la herramienta de Swagger Codegen.

- Swagger Codegen: https://swagger.io/tools/swagger-codegen/

2.3. Ducumentar utilizando Swagger-UI

La documentación de las APIs desarrolladas deberán hacerse públicas mediante el uso de Swagger-UI.

Existen varias formas de incluir Swagger-UI en los desarrollos, pero si se realiza a través de Plugins que autogeneren la documentación, debe verificarse que coincida exactamente con la especificación definida, pues algunos plugins se encuentran desactualizados o no soportan las últimas versiones de la especificación OpenAPI.

Es por esto que desde la OT-I se aconseja la exposición de Swagger-UI directamente desde el documento de definición tal y como muestra el **apartado 5.3.**

3. ¿QUÉ ES OPENAPI?

OpenAPI es una versión estandarizada de la especificación Swagger que proporciona un modo independiente del idioma de presentar API REST.

Entre los aspectos de la API que permite definir, destacamos:

- Parámetros de la API: versión de OpenAPI, información de la API REST (título, descripción y versión)
- URL base para la API en cada uno de sus entornos.
- Autenticación para el acceso a la API: soporta métodos de autenticación HTTP simples (usuario/contraseña), como otros más robustos como el uso de tokens API key, Oauth 2 y OpenID.
- Esquemas comunes: definiciones genéricas que pueden ser usadas como peticiones y/o respuestas, útiles para la homogenización de los errores devueltos a contemplar, por ejemplo.
- Por cada recurso:
  - Operaciones HTTP (GET, POST, PUT, PATCH, DELETE): Se identifican con cada uno de los servicios publicados que permiten interactuar con los recursos.
  - Parámetros del servicio: Parámetros a contemplar en las cabeceras HTTP o en la URL del recurso (pathParam, queryParam):
    - pathParam: Parámetros obligatorios que componen la URL completa de acceso al recurso.
    - queryParam: Parámetros obligatorios u opcionales utilizados para filtrar la respuesta a la consulta de un recurso, por lo que están ligados a operaciones GET.
  - Request Body: Esquema del mensaje de entrada que debe contemplar el cliente para la invocación del servicio.
  - Response Body: Esquemas de mensajes de salida que puede devolver el servidor tras la invocación del servicio. Se contemplan tanto los mensajes correctamente procesados (HTTP 2XX) como los erróneos (HTTP 3XX, 4XX, 5XX).

4. BENEFICIOS DE UTILIZAR OPENAPI

La especificación OpenAPI (OAS) es a REST como el WSDL a SOAP. Establece un marco común para diseñadores, desarrolladores y testers sobre como construir y mantener APIs.

OAS es agnóstico en cuanto al lenguaje se refiere y está pensado para ser legible tanto por personas como por máquinas. Esto permite conocer el funcionamiento y capacidades de un servicio sin necesidad de acceder al código fuente, revisar documentación adicional o inspeccionar el tráfico de red realizando pruebas.

Debido a la gran repercusión que ha tenido esta especificación, también existen multitud de herramientas que nos facilitarán las tareas de exponer la documentación, generar el código fuente de la aplicación servidora y generar los clientes que consumen la API en multitud de lenguajes de programación.

Para aprovechar correctamente las ventajas del uso de OpenAPI, se aconseja que el ciclo de desarrollo comience por la fase de diseño (Contract-First). Esto significa que antes de empezar a construir la API, pensar en la lógica de negocio o centrarse en posibles errores o defectos diseñaremos la interfaz de la API detallando las peticiones y respuestas posibles.

Empezar el ciclo de desarrollo por la definición nos aporta ventajas como las siguientes:

- Permite centrarse en las necesidades del consumidor de la API abstraído de los problemas que puedan surgir durante el desarrollo.
- Reduce la dependencia entre equipos que tienen que trabajar con la API a desarrollar y comenzar con antelación los trabajos de estos conociendo desde un principio la funcionalidad exacta del servicio.
- Debido a eliminar la dependencia entre equipos, el tiempo de lanzamiento de la API es considerablemente reducido.

5. CONSTRUYENDO UNA API REST SIGUIENDO LA ESPECIFICACIÓN DE OPENAPI

La especificación OpenAPI inicialmente era parte de Swagger y propiedad de SmartBear. En 2015, SmartBear donó esta especificación a la OpenAPI Initiative, un proyecto colaborativo perteneciente a la Linux Foundation. Dentro de la Linux Foundation existe un comité encargado de la evolución de la especificación llamado también OpenAPI.

El detalle de la especificación puede ser consultado en la siguiente dirección.

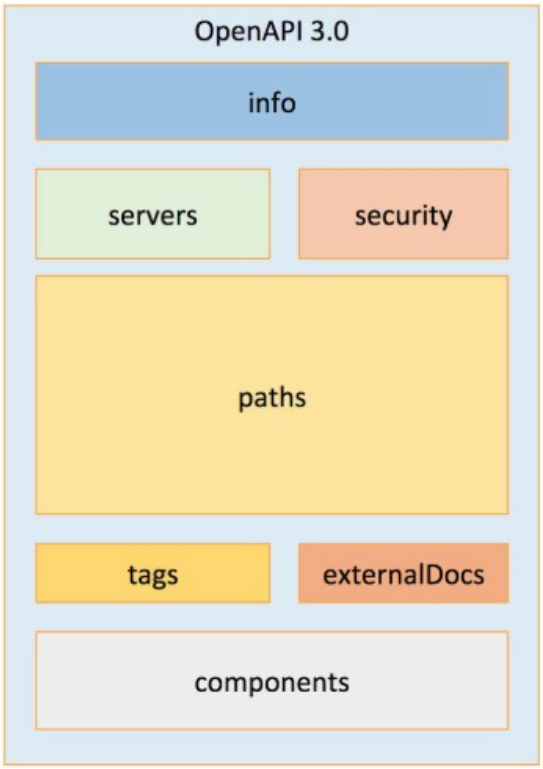
http://spec.openapis.org/oas/v3.0.3

A continuación se detalla como proceder para el desarrollo de una API partiendo de su definición y se aconseja el uso de determinadas herramientas que facilitarán la tarea.

5.1. Definición de la API

Antes de comenzar con los desarrollos, es necesario diseñar la API. En este paso es donde se hará uso de las especificaciones propuestas por OpenAPI y se creará el documento en formato YAML o JSON donde quedará definida la API.

La definición de una API para OpenAPI en su versión 3.0 se compone de la siguiente estructura de objetos:



- OpenAPI:(Obligatorio) (String) Declara la versión de la especificación OpenAPI utilizada.

```
openapi: 3.0.0
```

- Info: (Obligatorio) (Info Object) Objeto que contiene los metadatos de la API.

  http://spec.openapis.org/oas/v3.0.3.html#infoObject

```yaml
info:
  title: Ejemplo OTI-CHIE
  version: 1.0.0
  description: Servidor de ejemplo de uso de OpenAPI. Desarrollado en
    SpringBoot
  contact:
    email: l-interoperabilidad.chie@juntadeandalucia.es
```

- Servers: ([Server Object]) Array de objetos que representan al servidor donde se encuentra alojada la API. En caso de no definirlo, establece por defecto la URL a "/".
  http://spec.openapis.org/oas/v3.0.3.html#serverObject

```yaml
servers:
  - url: 'http://localhost:8090'
```

- Security:([Security Requirement Object]) Array de objetos que definen el esquema de seguridad utilizado en la API, puede definirse a nivel global o a nivel operacional

  http://spec.openapis.org/oas/v3.0.3.html#securityRequirementObject

```yaml
security:
  - basicAuth: []  # basicAuth es un esquema definido en Components
```

- Paths: (Obligatorio) ([Paths Object]) Array de objetos que definen las distintas rutas para los endpoints y las operaciones construyendo de esta forma la URL completa. Las URL completas queda compuestas por la URL del servidor concatenado con el PATH del recurso.

  En cada ruta hemos de definir las diferentes operaciones HTTP (GET, POST, PUT ...) y para cada operación, definir al menos el formato de salida de la operación (application/json, application/xml), parámetros de entrada y posibles respuestas.

  http://spec.openapis.org/oas/v3.0.3.html#pathsObject

```yaml
paths:
  /lista:
    get:
      tags:
        - Objetos OpenAPI
      operationId: listadoObjetos
      summary: Lista Objetos OpenAPI
      responses:
        '500':
          description: Server Error
          content:
            application/json:
              examples:
                ServerError:
                  value:
                    timestamp: '2020-04-29T16:14:14.048Z'
                    status: 500
                    error: Internal Server Error
                    exception: java.lang.ClassCastException
                    message: java.lang.String cannot be cast to net.minidev
                      .json.JSONObject
                    path: /objeto
        '200':
          description: 200 response
```

- Tags: ([Tag Object]) Array de objetos que permiten agrupar las operaciones con distintas etiquetas:

  http://spec.openapis.org/oas/v3.0.3.html#tagObject

```yaml
tags:
  - name: Tag 1
  - name: Tag 2
```

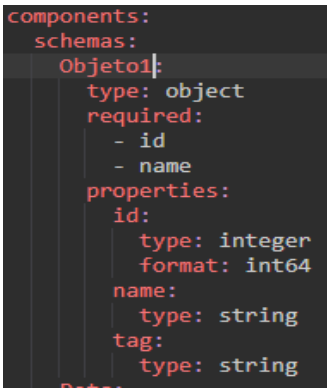
- ExternalDocs: (External Documentation Object) Objeto que permite hacer referencia a documentación externa mas extendida en caso de ser necesario.

  http://spec.openapis.org/oas/v3.0.3.html#externalDocumentationObject

```yaml
externalDocs:
  description: OpenAPI Specification
  url: http://spec.openapis.org/oas/v3.0.3.html
```

- Components: (Components Object) Objeto que permite evitar duplicidad en las declaraciones de las distintas operaciones o recursos mediante la definición de esquemas para respuestas comunes o parámetros que luego pueden ser referenciados en las operaciones.

  http://spec.openapis.org/oas/v3.0.3.html#componentsObject



Con el objetivo de facilitar el diseño de la API, podemos hacer uso de Swagger Editor, descargando la herramienta o haciendo uso de su versión Online https://editor.swagger.io/ .

5.2. Implementación de la API

Una vez definida la API, se procederá con su implementación en el lenguaje que se considere apropiado. Una vez más las herramientas ofrecidas por SmartBear y OpenAPI facilitan la tarea.

Para la implementación, se puede hacer uso de la herramienta Swagger Codegen o OpenAPI Generator, estas generarán el esqueleto de las aplicaciones (tanto clientes como servidor) a partir de la definición de la API y soportando multitud de lenguajes.

- Swagger Codegen: https://swagger.io/tools/swagger-codegen/
- OpenAPI Generator: https://github.com/OpenAPITools/openapi-generator

5.3. Documentando la API con Swagger-UI

En multitud de lenguajes y si hemos generado el esqueleto de la aplicación mediante una de las herramientas mencionadas anteriormente, no será necesario realizar un esfuerzo extra para publicar nuestra documentación en la web, puesto que ya incluyen plugins o dependencias encargados de auto-publicarla.

Puede darse el caso que para el lenguaje deseado, la dependencia encargada de publicar la definición de la API no soporte las últimas versiones de la especificación de OpenAPI, por lo que a continuación aconsejamos una forma sencilla de publicar la interfaz de Swagger en nuestra aplicación que nos asegura exponer exactamente lo definido y no algo auto generado a partir del código.

Para esto usaremos la herramienta de Swagger-UI, la cual puede ser obtenida de:

https://github.com/swagger-api/swagger-ui

La carpeta "dist" incluye lo necesario para ser publicada en la aplicación, tan solo es necesario modificar el fichero "index.html" y relacionarlo con la URL o ruta del fichero de especificación.

Admite los formatos JSON y YAML.

```
window.onload = function() {
  // Begin Swagger UI call region
  const ui = SwaggerUIBundle({
    url: './apiDefinition.json',
    dom_id: '#swagger-ui',
    deepLinking: true,
    presets: [
      SwaggerUIBundle.presets.apis,
      SwaggerUIStandalonePreset
    ],
```

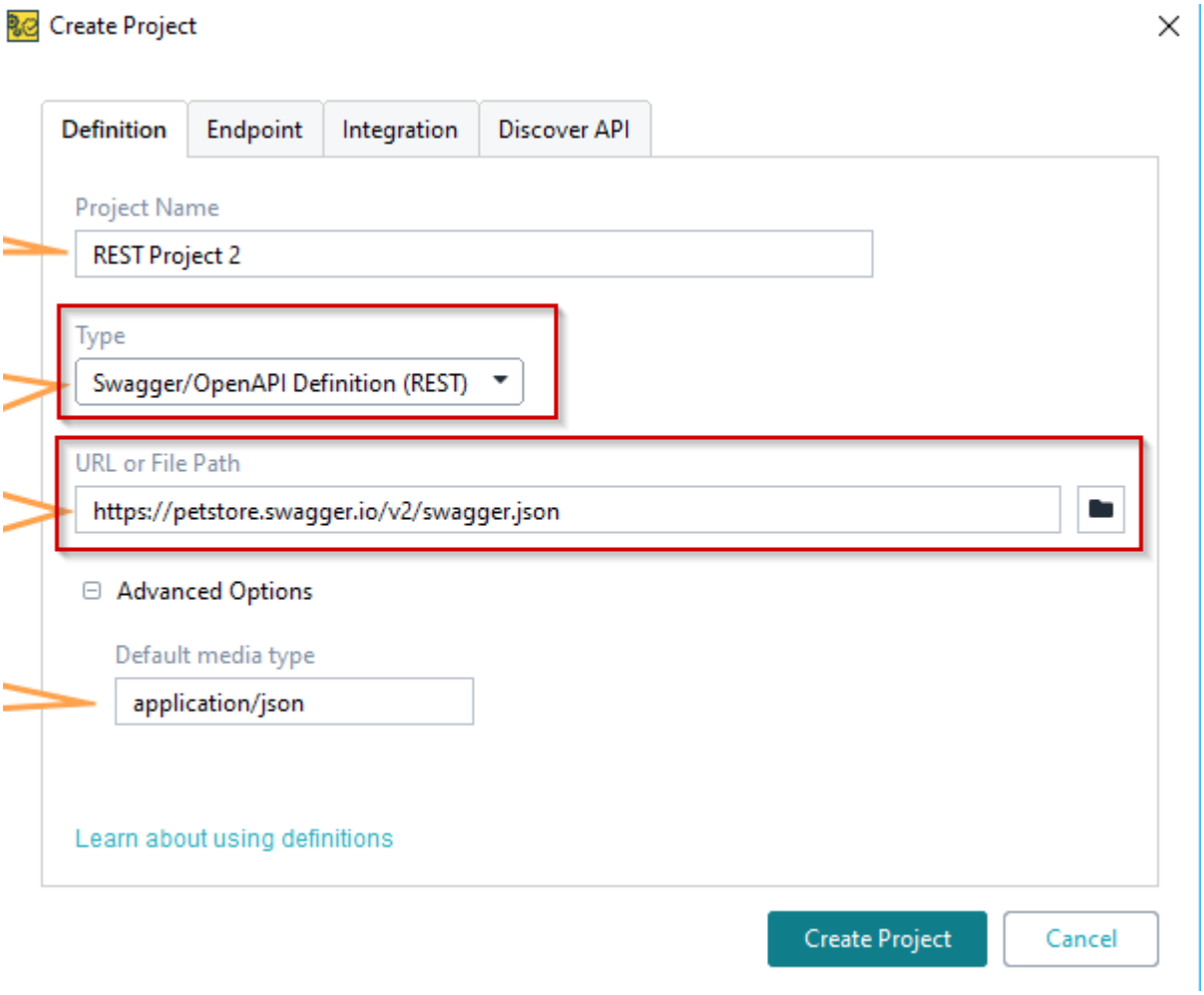
5.4. Testing

Implementar la API siguiendo las especificaciones propuestas por OpenAPI también facilita la tarea de testing.

A partir de la definición realizada en el fichero YAML o JSON, es posible usar Swagger Inspector como herramienta de testing para la API.

https://swagger.io/tools/swagger-inspector/

Otra alternativa es utilizar SOAP-UI, en el cual también es posible generar un proyecto a partir de la definición de la API.


Create Project
Definition
Endpoint
Integration
Discover API
Project Name
REST Project 2
Type
Swagger/OpenAPI Definition (REST)
URL or File Path
https://petstore.swagger.io/v2/swagger.json
Advanced Options
Default media type
application/json
Learn about using definitions
Create Project
Cancel

5.5. Ejemplo Spring Boot

Acorde con esta guia, se ha desarrollado una API básica como ejemplo en Spring Boot.

Para la generación de esta API se hace uso de Swagger Codegen para generar el esqueleto de la aplicación a partir de la definición, aunque dado que incluye la dependencia de Springfox por defecto para publicar la documentación y esta no es compatible con la versión 3 de la especificación de OpenAPI, se ha eliminado dicha dependencia y procedido a publicar la documentación como se explica en el apartado 4.3.

Proporcionamos junto a este documento y a modo de apoyo tanto el código fuente del desarrollo como el código compilado y empaquetado en un jar.

Para ejecutar el jar, usaremos el siguiente comando:

java -jar OTI-CHIE_OpenAPI_Example-1.0.0.jar

Esto ejecutará la aplicación y la tendremos disponible en la dirección:

http://localhost:8090/