# OpenAPI

## Introducción de OpenAPI

La iniciativa OpenAPI (OAI) fue creada por un consorcio de expertos de la industria que reconocen el enorme valor de estandarizar la forma en que se describen las API. Como una estructura de gobernanza abierta bajo la Fundación Linux, la OAI se centra en crear, evolucionar y promover un formato de descripción neutral al proveedor. La especificación OpenAPI se basó originalmente en la especificación Swagger, donada por SmartBear Software.

## Especificación OpenAPI

La especificación OpenAPI define un formato de archivo para describir las API RESTful. Una descripción de API OpenAPI permite a cualquier cliente entender y consumir la funcionalidad de una API sin necesidad de acceder al código fuente, documentación adicional o esquemas de interfaz. El formato de archivo OpenAPI es un formato de archivo YAML y se puede representar como JSON. El formato de archivo OpenAPI se puede usar para describir cualquier tipo de API RESTful, pero no se limita a ellas. También se puede usar para describir APIs RPC o de acceso a datos.

## Herramientas de OpenAPI

La OAI ha creado una serie de herramientas para ayudar a los desarrolladores a crear, documentar y consumir APIs.

*Estas herramientas incluyen:*

- Swagger Editor: un editor de código abierto basado en navegador para diseñar y documentar APIs.
- Swagger UI: una interfaz de usuario de código abierto basada en navegador para visualizar documentación de API.
- Swagger Codegen: un generador de código de código abierto para generar clientes y servidores de API a partir de una descripción de API OpenAPI.
- Swagger Inspector: una herramienta de código abierto para probar y documentar APIs.
- SwaggerHub: una plataforma de código abierto para diseñar, documentar, probar y compartir APIs.

## Instalación de OpenAPI

Es posible trabajar con OpenAPI de forma local o en la nube. Para trabajar con OpenAPI de forma local, se debe instalar la herramienta Swagger Editor. Para trabajar con OpenAPI en la nube, se debe crear una cuenta en SwaggerHub.

*Cuando se describe una API, se define:*

- Los recursos disponibles y sus operaciones

- Los parámetros de operación y sus valores
- Los modelos de datos de respuesta
- Los modelos de datos de entrada
- Los esquemas de seguridad

# Apartados de la especificación OpenAPI

## Tipos de datos

*OpenAPI define los siguientes tipos de datos:*

- int32
- int64
- float
- double
- boolean
- string
  - byte
  - binary
  - date
  - dateTime
  - password

# Campos de información

*OpenAPI define los siguientes campos de información:*

- openapi: Versión de la especificación que se está usando para describir la API. La versión actual es 3.0.3.
- info: Información general sobre la API.
- servers: Una lista de servidores que se pueden usar para ejecutar la API.
- paths: La lista de rutas de la API.
- components: Los esquemas de componentes reutilizables para las operaciones.
- security: La lista de esquemas de seguridad que se pueden usar para la API.
- tags: La lista de etiquetas que se pueden usar para la API.
- externalDocs: Información adicional sobre la API.

## Objeto info

*El objeto info contiene información general sobre la API. La información incluye:*

- title: El nombre de la API.
- description: Una descripción de la API.
- termsOfService: Una URL a los términos de servicio de la API.
- contact: Información de contacto para la API.
- license: Información sobre la licencia de la API.
- version: La versión de la API.

*Ejemplo de objeto info:*

```
title: Sample Pet Store App
description: This is a sample server for a pet store.
termsOfService: http://example.com/terms/
contact:
  name: API Support
  url: http://www.example.com/support
  email: example@example.com
license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
version: 1.0.1
```

## Objeto contact

*El objeto contact contiene información de contacto para la API. La información incluye:*

- name: El nombre de la persona de contacto para la API.
- url: La URL de la persona de contacto para la API.
- email: El correo electrónico de la persona de contacto para la API.

*Ejemplo de objeto contact:*

```
contact:
  name: API Support
  url: http://www.example.com/support
  email: contact@example.com
```

## Objeto license

*El objeto license contiene información sobre la licencia de la API. La información incluye:*

- name: El nombre de la licencia de la API.
- url: La URL de la licencia de la API.

*Ejemplo de objeto license:*

```
license:
  name: Apache 2.0
  url: http://www.apache.org/licenses/LICENSE-2.0.html
```

## Objeto server

*El objeto server contiene información sobre un servidor que se puede usar para ejecutar la API. La información incluye:*

- url: La URL del servidor.
- description: Una descripción del servidor.
- variables: Una lista de variables que se pueden usar para definir el valor de la URL del servidor.

*Ejemplo de objeto server:*

```
servers:
- url: https://{username}.gigantic-server.com:{port}/{basePath}
  description: The production API server
  variables:
    username:
      # note! no enum here means it is an open value
      default: demo
      description: this value is assigned by the service provider, in this example `gigantic-
server.com`
    port:
      enum:
        - '8443'
        - '443'
      default: '8443'
    basePath:
      # open meaning there is the opportunity to use special base paths as assigned by the
provider, default is `v2`
      default: v2
```

## Objeto variable de servidor

*El objeto variable de servidor contiene información sobre una variable que se puede usar para definir el valor de la URL del servidor. La información incluye:*

- enum: Una lista de valores posibles para la variable.
- default: El valor predeterminado para la variable.
- description: Una descripción de la variable.

*Ejemplo de objeto variable de servidor:*

```
servers:
- url: https://{username}.gigantic-server.com:{port}/{basePath}
  description: The production API server
  variables:
    username:
      # note! no enum here means it is an open value
      default: demo
      description: this value is assigned by the service provider, in this example `gigantic-
server.com`
    port:
      enum:
        - '8443'
        - '443'
      default: '8443'
    basePath:
      # open meaning there is the opportunity to use special base paths as assigned by the
provider, default is `v2`
      default: v2
```

## Objeto componentes

*El objeto componentes contiene los esquemas de componentes reutilizables para las operaciones. La información incluye:*

- schemas: Una lista de esquemas de componentes reutilizables para las operaciones.
- responses: Una lista de respuestas de componentes reutilizables para las operaciones.
- parameters: Una lista de parámetros de componentes reutilizables para las operaciones.
- examples: Una lista de ejemplos de componentes reutilizables para las operaciones.
- requestBodies: Una lista de cuerpos de solicitud de componentes reutilizables para las operaciones.
- headers: Una lista de encabezados de componentes reutilizables para las operaciones.
- securitySchemes: Una lista de esquemas de seguridad de componentes reutilizables para las operaciones.
- links: Una lista de enlaces de componentes reutilizables para las operaciones.
- callbacks: Una lista de devoluciones de llamada de componentes reutilizables para las operaciones.

*Ejemplo de objeto componentes:*

```
components:
  schemas:
    Category:
      type: object
      properties:
        id:
          type: integer
          format: int64
        name:
          type: string
    Tag:
      type: object
      properties:
        id:
          type: integer
          format: int64
        name:
          type: string
  parameters:
    skipParam:
      name: skip
      in: query
      description: number of items to skip
      required: true
      schema:
        type: integer
        format: int32
    limitParam:
      name: limit
      in: query
      description: max records to return
      required: true
      schema:
        type: integer
        format: int32
  responses:
    NotFound:
      description: Entity not found.
    IllegalInput:
      description: Illegal input for operation.
    GeneralError:
      description: General Error
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/GeneralError'
  securitySchemes:
    api_key:
      type: apiKey
      name: api_key
      in: header
    petstore_auth:
      type: oauth2
      flows:
        implicit:
          authorizationUrl: http://example.org/api/oauth/dialog
          scopes:
            write:pets: modify pets in your account
            read:pets: read your pets
  examples:
```

## Objeto paths

*El objeto paths contiene la lista de rutas y operaciones disponibles en la API. La información incluye:*

- método: Una operación GET, PUT, POST, DELETE, OPTIONS, HEAD, PATCH, TRACE o CONNECT.
  - description: Una descripción de la operación.
  - responses: Una lista de respuestas de la operación.
    - código de respuesta: Una respuesta de la operación.
      - description: Una descripción de la respuesta.
      - content: Una lista de contenidos de la respuesta.
        - media type: Un tipo de medio de la respuesta.
          - schema: Un esquema de la respuesta.
          - type: Un tipo de la respuesta.

*Ejemplo de objeto paths:*

```
paths:
  /pets:
    get:
      description: Returns all pets from the system that the user has access to
      responses:
        '200':
          description: A list of pets.
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Pet'
    post:
      description: Creates a new pet in the store.  Duplicates are allowed
      responses:
        '200':
          description: Null response
  /pets/{id}:
    get:
      description: Returns a user based on a single ID, if the user does not have access to
the pet
      responses:
        '200':
          description: Expected response to a valid request
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Pet'
        default:
          description: unexpected error
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorModel'
```

## Objeto item del objeto paths

*El objeto item del objeto paths contiene la lista de rutas y operaciones disponibles en la API. La información incluye:*

- método: Una operación GET, PUT, POST, DELETE, OPTIONS, HEAD, PATCH, TRACE o CONNECT.
  - description: Una descripción de la operación.
  - summary: Un resumen de la operación.
  - operationId: Un identificador de la operación.
  - responses: Una lista de respuestas de la operación.
    - código de respuesta: Una respuesta de la operación.
      - description: Una descripción de la respuesta.
      - content: Una lista de contenidos de la respuesta.
    - default: Una respuesta por defecto de la operación.
      - description: Una descripción de la respuesta por defecto.
      - content: Una lista de contenidos de la respuesta por defecto.
  - parameters: Una lista de parámetros de la operación.
    - name: Un nombre del parámetro.
    - in: Un lugar del parámetro: query, header, path o cookie.
    - description: Una descripción del parámetro.
    - required: Un valor booleano que indica si el parámetro es requerido.
    - schema: Un esquema del parámetro.
    - style: Un estilo del parámetro.

*Ejemplo de objeto item del objeto paths:*

```
paths:
  /pets:
    get:
      description: Returns all pets from the system that the user has access to
      responses:
        '200':
          description: A list of pets.
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Pet'
    post:
      description: Creates a new pet in the store.  Duplicates are allowed
      responses:
        '200':
          description: Null response
  /pets/{id}:
    get:
      description: Returns a user based on a single ID, if the user does not have access to
the pet
      responses:
        '200':
          description: Expected response to a valid request
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Pet'
        default:
          description: unexpected error
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorModel'
```

## Objeto operation

*El objeto operation contiene la lista de rutas y operaciones disponibles en la API. La información incluye:*

- tags: Una lista de etiquetas de la operación.
- summary: Un resumen de la operación.
- description: Una descripción de la operación.
- externalDocs: Una lista de documentos externos de la operación.
- operationId: Un identificador de la operación.
- parameters: Una lista de parámetros de la operación.
- requestBody: Una lista de solicitudes de la operación.
- responses: Una lista de respuestas de la operación.
- callbacks: Una lista de callbacks de la operación.
- deprecated: Un valor booleano que indica si la operación está obsoleta.
- security: Una lista de seguridad de la operación.
- servers: Una lista de servidores de la operación.

*Ejemplo de objeto operation:*

```
paths:
  /pets:
    get:
      description: Returns all pets from the system that the user has access to
      responses:
        '200':
          description: A list of pets.
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Pet'
    post:
      description: Creates a new pet in the store.  Duplicates are allowed
      responses:
        '200':
          description: Null response
  /pets/{id}:
    get:
      description: Returns a user based on a single ID, if the user does not have access to
the pet
      responses:
        '200':
          description: Expected response to a valid request
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Pet'
        default:
          description: unexpected error
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorModel'
```

## Objeto de documentación externa

*El objeto de documentación externa contiene la lista de documentos externos de la operación. La información incluye:*

- description: Una descripción del documento externo.
- url: Una URL del documento externo.

*Ejemplo de objeto de documentación externa:*

```
description: Find more info here
url: https://swagger.io/about
```

## Objeto de parámetro

*El objeto de parámetro contiene la lista de parámetros de la operación. La información incluye:*

- name: Un nombre del parámetro.
- in: Un lugar del parámetro: query, header, path o cookie.
- description: Una descripción del parámetro.
- required: Un valor booleano que indica si el parámetro es requerido.
- deprecated: Un valor booleano que indica si el parámetro está obsoleto.
- allowEmptyValue: Un valor booleano que indica si el parámetro permite un valor vacío.

*Ejemplo de objeto de parámetro:*

```
parameters:
  - name: id
    in: path
    description: ID of pet to use
    required: true
    schema:
      type: array
      items:
        type: string
  - name: id
    in: path
    description: ID of pet to use
    required: true
    schema:
      type: array
      items:
        type: string
  - name: id
    in: path
    description: ID of pet to use
    required: true
    schema:
      type: array
      items:
        type: string
```

## Objeto de solicitud

*El objeto de solicitud contiene la lista de solicitudes de la operación. La información incluye:*

- description: Una descripción de la solicitud.
- content: Una lista de contenidos de la solicitud.
- required: Un valor booleano que indica si la solicitud es requerida.

*Ejemplo de objeto de solicitud:*

```
requestBody:
  description: Pet to add to the store
  content:
    '*/*':
      schema:
        $ref: '#/components/schemas/NewPet'
  required: true
```

## Objeto de respuesta

*El objeto de respuesta contiene la lista de respuestas de la operación. La información incluye:*

- description: Una descripción de la respuesta.
- headers: Una lista de encabezados de la respuesta.
- content: Una lista de contenidos de la respuesta.
- links: Una lista de enlaces de la respuesta.

*Ejemplo de objeto de respuesta:*

```
responses:
  '200':
    description: pet response
    content:
      '*/*':
        schema:
          $ref: '#/components/schemas/Pet'
  default:
    description: unexpected error
    content:
      '*/*':
        schema:
          $ref: '#/components/schemas/ErrorModel'
```

## Objeto de callback

*El objeto de callback contiene la lista de callbacks de la operación. La información incluye:*

- '{$request.body#/callbackUrl}': Una lista de callbacks de la operación.

*Ejemplo de objeto de callback:*

```
callbacks:
  '{$request.body#/callbackUrl}':
    post:
      requestBody:
        description: Callback payload
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Status'
      responses:
        '200':
          description: Callback response
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/InlineResponse200'
```

## Objeto de ejemplo

*El objeto de ejemplo contiene la lista de ejemplos de la operación. La información incluye:*

- summary: Un resumen del ejemplo.
- description: Una descripción del ejemplo.
- value: Un valor del ejemplo.
- externalValue: Un valor externo del ejemplo.

*Ejemplo de objeto de ejemplo:*

```
examples:
  user:
    summary: User example
    description: User example in JSON
    value:
      username: foo
      password: bar
      id: 1
  user:
    summary: User example
    description: User example in JSON
    value:
      username: foo
      password: bar
      id: 1
  user:
    summary: User example
    description: User example in JSON
    value:
      username: foo
      password: bar
      id: 1
```

## Objeto de enlace

*El objeto de enlace contiene la lista de enlaces de la operación. La información incluye:*

- operationRef: Una referencia de la operación.
- operationId: Un ID de la operación.
- parameters: Una lista de parámetros de la operación.
- requestBody: Una solicitud de la operación.
- description: Una descripción de la operación.
- server: Un servidor de la operación.

*Ejemplo de objeto de enlace:*

```
links:
  UserRepositories:
    # the relative URL of the target operation
    operationRef: '#/paths/~1repositories~1{username}/get'
    # a literal value, to be used as a request body when calling the target operation
    operationId: getRepositories
    parameters:
      username: '{$request.path.username}'
    # alternative server array to service this operation
    requestBody: '{$request.body}'
    description: The repositories owned by the user.
    server:
      url: https://gigantic-server.com/
      description: The production API server
```

## Objeto Tag

El objeto Tag contiene la lista de tags de la operación que se pueden usar para agrupar y organizar las operaciones.

*este objeto incluye los siguientes atributos:*

- name: Un nombre del tag.
- description: Una descripción del tag.
- externalDocs: Una lista de documentos externos del tag.

*Ejemplo de objeto Tag:*

```
tags:
  - name: pet
    description: Everything about your Pets
    externalDocs:
      description: Find out more
      url: 'http://swagger.io'
  - name: pet
    description: Everything about your Pets
    externalDocs:
      description: Find out more
      url: 'http://swagger.io'
  - name: pet
    description: Everything about your Pets
    externalDocs:
      description: Find out more
      url: 'http://swagger.io'
```

## Objeto de esquema

El objeto de esquema contiene la lista de esquemas que se pueden usar para definir las peticiones y las respuesta de los mensajes que se usan en el API.

*Los atributos de este objeto incluyen:*

- type: Un tipo de esquema.
- properties: Una lista de propiedades de esquema.
- required: Una lista de requerimientos de esquema.
- example: Un ejemplo de esquema.

*Ejemplo de objeto de esquema:*

```
schemas:
  NewPet:
    type: object
    required:
      - name
    properties:
      name:
        type: string
      tag:
        type: string
    example:
      name: doggie
      tag: puppy
```

## Objeto de seguridad

El objeto de seguridad contiene la lista de esquemas de seguridad que se pueden usar para definir los esquemas de seguridad que se usan en el API.

*Los atributos de este objeto incluyen:*

- type: Un tipo de esquema de seguridad.
- description: Una descripción de esquema de seguridad.
- name: Un nombre de esquema de seguridad.
- in: Un lugar de esquema de seguridad.
- scheme: Un esquema de esquema de seguridad.
- bearerFormat: Un formato de esquema de seguridad.
- flows: Un flujo de esquema de seguridad.
- openIdConnectUrl: Una URL de esquema de seguridad.

*Ejemplo de objeto de seguridad:*

```
securitySchemes:
  ApiKeyAuth:
    type: apiKey
    description: 'Standard Authorization header using the Bearer scheme. Example:
"Authorization: Bearer {token}"'
    name: Authorization
    in: header
    scheme: bearer
    bearerFormat: JWT
    flows:
      implicit:
        authorizationUrl: 'https://example.com/api/oauth/dialog'
        scopes:
          write:pets: modify pets in your account
          read:pets: read your pets
    openIdConnectUrl: 'https://example.com/.well-known/openid-configuration'
```