



ISWD613 APLICACIONES WEB

NOMBRE ESTUDIANTE(S): Luis Andrés Guerrero Hinojosa
FECHA: 10/11/2025

TEMA: Examen 01

Introducción

El presente examen tiene como objetivo demostrar la capacidad de documentar APIs RESTful existentes utilizando la especificación OpenAPI 3.0 mediante Swagger. Para ello, se ha seleccionado JSONPlaceholder, una API REST falsa ampliamente utilizada en el ámbito educativo y de pruebas de desarrollo web. Esta API proporciona seis recursos principales (posts, comments, albums, photos, todos y users) con sus respectivas operaciones CRUD, lo que permite ejemplificar de manera integral los conceptos fundamentales de la arquitectura REST. La documentación generada no solo describe técnicamente cada endpoint disponible, sino que también establece un estándar profesional de comunicación entre desarrolladores frontend y backend, facilitando la integración y el mantenimiento de sistemas distribuidos. Este ejercicio representa una competencia esencial en el desarrollo de aplicaciones web modernas, donde la documentación precisa y estructurada de interfaces de programación es un requisito indispensable para garantizar la escalabilidad, mantenibilidad y colaboración efectiva en equipos de desarrollo.

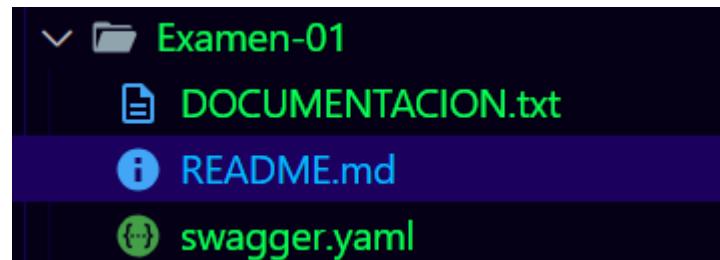
Desarrollo

La documentación de la API JSONPlaceholder se estructuró siguiendo rigurosamente la especificación OpenAPI 3.0.0, implementando un archivo YAML que contempla todos los componentes esenciales de una especificación profesional. Se definió la información básica de la API incluyendo título, descripción, versión y datos de contacto, estableciendo el servidor base en <https://jsonplaceholder.typicode.com>. La organización de los endpoints se realizó mediante un sistema de etiquetas que agrupa lógicamente los recursos en seis categorías principales: Posts, Comments, Albums, Photos, Todos y Users. Para cada recurso se documentaron exhaustivamente sus operaciones CRUD correspondientes, totalizando veintiséis endpoints que abarcan operaciones de lectura (GET), creación (POST), actualización completa (PUT), actualización parcial (PATCH) y eliminación (DELETE). Cada endpoint incluye descripciones detalladas, especificación de parámetros de ruta cuando corresponde, definición de cuerpos de petición con sus respectivos tipos de contenido, y documentación completa de respuestas HTTP con sus códigos de estado apropiados (200, 201, 204, 404). La sección de componentes define esquemas reutilizables para cada entidad, diferenciando entre esquemas de lectura que incluyen el identificador único y esquemas de escritura utilizados en operaciones de creación y actualización. Esta estructura modular mediante referencias (\$ref) garantiza la consistencia en toda la documentación y facilita el mantenimiento futuro. Adicionalmente, se documentaron las relaciones entre recursos mediante endpoints anidados, como la obtención de comentarios específicos de un post (/posts/{id}/comments) o los álbumes de un



ISWD613 APPLICACIONES WEB

usuario (/users/{id}/albums), demostrando comprensión de patrones avanzados de diseño de APIs RESTful. La documentación resultante puede ser visualizada interactivamente en herramientas como Swagger Editor o Swagger UI, permitiendo no solo la consulta de especificaciones sino también la ejecución de pruebas directamente desde la interfaz de documentación.



En swagger



ISWD613 APLICACIONES WEB

The screenshot shows the Swagger Editor interface with the following details:

- Left Panel (Code View):** Displays the OpenAPI 3.0 specification for the JSONPlaceholder API, including paths for /posts, /comments, /albums, /photos, and /users.
- Right Panel (Documentation View):** Shows the generated API documentation for the JSONPlaceholder API. It includes sections for each resource type (Posts, Comments, Albums, Photos, Users) with their respective operations (GET, POST, PUT, DELETE).
- Bottom Bar:** Shows the Windows taskbar with various application icons.

The screenshot shows the Swagger Editor interface with the following details:

- Left Panel (Code View):** Displays the same OpenAPI 3.0 specification for the JSONPlaceholder API as the first screenshot.
- Right Panel (Documentation View):** Shows the generated API documentation for the JSONPlaceholder API, organized into sections: Posts, Comments, Albums, Photos, and Users. Each section contains its corresponding API operations.
- Bottom Bar:** Shows the Windows taskbar with various application icons.



ISWD613 APLICACIONES WEB

The screenshot shows the Swagger Editor interface with the following details:

- Header:** Shows the URL `editor.swagger.io`.
- Toolbar:** Includes standard browser controls like back, forward, search, and refresh.
- Code Editor:** Displays the OpenAPI specification (JSON/YAML) for the API.
- Operations:** A list of operations grouped by resource.
 - Todos:** Operations related to tasks.
 - GET /todos:** Obtener todas las tareas.
 - POST /todos:** Crear una nueva tarea.
 - PUT /todos/{id}:** Actualizar una tarea por ID.
 - Users:** Operations related to users.
 - GET /users:** Obtener todos los usuarios.
 - GET /users/{id}:** Obtener usuario por ID.
- Schemas:** Definitions for data structures.
 - Post:** A schema for a task entry with properties `userId*`, `id*`, `title*`, and `body*`.
 - Posting:** A schema for a task update with properties `title*` and `body*`.

The screenshot shows the Swagger Editor interface with the following details:

- Header:** Shows the URL `editor.swagger.io`.
- Toolbar:** Includes standard browser controls like back, forward, search, and refresh.
- Code Editor:** Displays the OpenAPI specification (JSON/YAML) for the API.
- Operations:** A list of operations grouped by resource.
 - Comment:** A schema for a comment entry with properties `postId*`, `id*`, `name*`, `email*`, and `body*`.
 - CommentInput:** A schema for a comment input with properties `name*`, `email*`, `body*`, and `postId*`.
 - Album:** A schema for an album entry with properties `userId*`, `id*`, and `title*`.
 - AlbumInput:** A schema for an album input with properties `title*` and `userId*`.
 - Photo:** A schema for a photo entry with properties `albumId*`, `id*`, `title*`, `url*`, and `thumbnailUrl*`.

Conclusiones

A realización de este examen ha permitido consolidar conocimientos fundamentales sobre documentación de APIs mediante Swagger/OpenAPI, una habilidad crítica en el contexto actual del desarrollo de software distribuido. La especificación generada cumple con estándares profesionales de la industria, proporcionando una referencia completa y técnicamente precisa de todos los endpoints disponibles en JSONPlaceholder. Se ha demostrado competencia en la estructuración de documentación API mediante la correcta implementación de etiquetas, esquemas reutilizables, parámetros, cuerpos de petición y respuestas HTTP, aspectos que constituyen la base de una

ISWD613 APPLICACIONES WEB

comunicación efectiva entre sistemas y equipos de desarrollo. La experiencia adquirida en este ejercicio trasciende el ámbito académico, siendo directamente aplicable a escenarios reales de desarrollo donde la documentación clara y estructurada de interfaces de programación determina la eficiencia de la integración entre microservicios, facilita la incorporación de nuevos desarrolladores a proyectos existentes, y reduce significativamente el tiempo de resolución de incidencias. Este trabajo representa un componente esencial del perfil profesional de un desarrollador frontend moderno, quien debe ser capaz no solo de consumir APIs documentadas, sino también de documentar sus propias interfaces siguiendo estándares reconocidos internacionalmente. La documentación Swagger generada constituye un activo valioso que puede ser utilizado como referencia para futuros proyectos de documentación de APIs RESTful, consolidando así las mejores prácticas aprendidas durante este curso.

❖ PROMPT GENERADOR

Examen 001 Crea documentación Swagger/OpenAPI 3.0 completa para la API JSONPlaceholder.

CONTEXTO:

JSONPlaceholder es una API REST falsa para testing y prototipado disponible en <https://jsonplaceholder.typicode.com>. Contiene 6 recursos principales con sus operaciones CRUD.

REQUISITOS:

1. ESTRUCTURA DEL PROYECTO:

- Carpeta: Examen-01
- 1 archivo swagger.yaml con especificación OpenAPI 3.0.0
- 1 archivo README.md con documentación técnica
- 1 archivo DOCUMENTACION.txt con enlace a Google Docs

2. INFORMACIÓN BÁSICA DEL SWAGGER:

- Título: "JSONPlaceholder API - Documentación Completa"
- Versión: 1.0.0
- Descripción detallada incluyendo lista de recursos con emojis
- Contacto y licencia MIT
- Servidor: <https://jsonplaceholder.typicode.com>

3. TAGS (6 categorías):

- Posts: Operaciones relacionadas con publicaciones
- Comments: Operaciones relacionadas con comentarios
- Albums: Operaciones relacionadas con álbumes
- Photos: Operaciones relacionadas con fotografías
- Todos: Operaciones relacionadas con tareas
- Users: Operaciones relacionadas con usuarios

4. ENDPOINTS A DOCUMENTAR (26 en total):



ISWD613 APLICACIONES WEB

POSTS (6 endpoints):

- GET /posts (listar todos)
- POST /posts (crear)
- GET /posts/{id} (obtener por ID)
- PUT /posts/{id} (actualizar completo)
- PATCH /posts/{id} (actualizar parcial)
- DELETE /posts/{id} (eliminar)

COMMENTS (2 endpoints):

- GET /comments (listar todos, con query param postId opcional)
- POST /comments (crear)

ALBUMS (3 endpoints):

- GET /albums (listar todos)
- POST /albums (crear)
- GET /albums/{id} (obtener por ID)

PHOTOS (1 endpoint):

- GET /photos (listar todas, con query param albumId opcional)

TODOS (3 endpoints):

- GET /todos (listar todas)
- POST /todos (crear)
- GET /todos/{id} (obtener por ID)

USERS (2 endpoints):

- GET /users (listar todos)
- GET /users/{id} (obtener por ID)

5. PARA CADA ENDPOINT INCLUIR:

- Tag apropiado
- Summary (resumen corto)
- Description (descripción detallada)
- operationId (nombre único)
- Parameters (path params y query params si aplica)
- RequestBody (para POST/PUT/PATCH con schema y ejemplo)
- Responses con códigos HTTP:
 - * 200 OK: Operación exitosa
 - * 201 Created: Recurso creado
 - * 404 Not Found: Recurso no encontrado
- Ejemplos de respuesta en JSON

6. SCHEMAS EN COMPONENTS (12 esquemas):

Post:

- userId: integer



ISWD613 APLICACIONES WEB

- id: integer
- title: string
- body: string

PostInput (sin id):

- title: string
- body: string
- userId: integer

Comment:

- postId: integer
- id: integer
- name: string
- email: string (format: email)
- body: string

CommentInput (sin id):

- name: string
- email: string
- body: string
- postId: integer

Album:

- userId: integer
- id: integer
- title: string

AlbumInput (sin id):

- title: string
- userId: integer

Photo:

- albumId: integer
- id: integer
- title: string
- url: string (format: uri)
- thumbnailUrl: string (format: uri)

Todo:

- userId: integer
- id: integer
- title: string
- completed: boolean

TodoInput (sin id):

- title: string
- completed: boolean



ISWD613 APLICACIONES WEB

- **userId: integer**

User (objeto complejo):

- **id: integer**
- **name: string**
- **username: string**
- **email: string**
- **address: \$ref Address**
- **phone: string**
- **website: string**
- **company: \$ref Company**

Address:

- **street: string**
- **suite: string**
- **city: string**
- **zipcode: string**
- **geo: \$ref Geo**

Geo:

- **lat: string**
- **lng: string**

Company:

- **name: string**
- **catchPhrase: string**
- **bs: string**

7. USAR \$REF PARA REUTILIZACIÓN:

- Referenciar schemas desde paths: **\$ref: '#/components/schemas/Post'**
- Referenciar schemas anidados en User: **\$ref: '#/components/schemas/Address'**

8. EJEMPLOS REALISTAS:

- Usar datos reales de JSONPlaceholder en los ejemplos
- Post: "sunt aut facere repellat provident"
- User: "Leanne Graham", username "Bret"
- Comment email: "Eliseo@gardner.biz"
- Photo URL: "<https://via.placeholder.com/600/92c952>"

9. PARÁMETROS:

- Path parameters con required: true, schema type: integer
- Query parameters opcionales (postId, albumId)
- Incluir examples en cada parámetro

10. REQUEST BODIES:

- Content-Type: application/json
- Usar schemas Input (sin id)



ISWD613 APLICACIONES WEB

- Incluir examples con datos realistas

11. README.MD DEBE INCLUIR:

- Título del examen
- Descripción de JSONPlaceholder
- Lista de recursos documentados
- Total de endpoints (26)
- Instrucciones para visualizar en Swagger Editor
- Estructura del archivo YAML
- Conceptos OpenAPI aplicados
- Información del estudiante

12. DOCUMENTACION.TXT:

- Enlace a Google Docs
- Descripción: Documentación Swagger JSONPlaceholder API
- Contenido: Introducción, Desarrollo, Conclusiones, Capturas

13. FORMATO YAML:

- Indentación de 2 espacios
- Usar | para descripciones multilínea
- Ordenar alfabéticamente properties en schemas
- Comentarios con # para separar secciones

14. BUENAS PRÁCTICAS:

- OperationId único para cada endpoint
- Descripciones claras y concisas
- Required fields especificados en schemas
- Formats apropiados (email, uri)
- Minimum: 1 para IDs en path params

15. AL FINALIZAR:

- Validar YAML en Swagger Editor
- Verificar que todos los endpoints funcionen
- Crear commit: "Examen-01: Documentacion Swagger completa JSONPlaceholder API"
- Subir a GitHub