**ESCUELA POLITÉCNICA NACIONAL**
**FACULTAD DE INGENIERÍA EN SISTEMAS**
**INGENIERIA EN SOFTWARE**

**Nombre:** Jhonny Moreira

**Fecha:** 13/12/2025

**Materia:** Aplicaciones Web

# Examen

## 📌 1. Introducción

El presente examen tiene como objetivo aplicar los conocimientos adquiridos en las clases 008-009 sobre consumo de APIs mediante la herramienta Bruno, realizando la documentación formal de los endpoints utilizados con Swagger (OpenAPI 3.0) para estandarizar y facilitar la comprensión de su uso, parámetros y respuestas.

La API documentada corresponde a JSONPlaceholder, un servicio público de prueba utilizado para simular recursos RESTful, útil para propósitos educativos y de testing sin necesidad de crear un backend real.

## 🎯 2. Objetivos del Examen

El examen propone:

Generar la documentación completa en formato OpenAPI (Swagger) de los endpoints utilizados en la práctica de clase (posts, comments, albums, photos, todos y users).

Permitir la visualización interactiva de dicha documentación mediante Swagger Editor y Swagger UI.

Incorporar la documentación generada en el repositorio bajo la carpeta Examen-01 para su evaluación.

## 📚 3. API Documentada

Se documentaron los siguientes recursos y métodos HTTP disponibles en JSONPlaceholder:

| Recurso | Método | Descripción |
|---|---|---|
| /posts | GET | Obtener todos los posts |
| /posts | POST | Crear un nuevo post |
| /posts/{id} | GET | Obtener post por ID |
| /posts/{id} | PUT | Actualizar post |
| /posts/{id} | PATCH | Modificar parcial de post |
| /posts/{id} | DELETE | Eliminar post |
| /comments | GET | Obtener todos los comentarios |
| /comments | POST | Crear comentario |

| Recurso | Método | Descripción |
|---|---|---|
| /albums | GET/POST | Obtener/crear álbum |
| /photos | GET | Obtener fotos |
| /todos | GET/POST | Obtener/crear tareas |
| /users | GET | Obtener usuarios |

*(Estos endpoints fueron trabajados en Bruno durante las clases 008-009)*

## 🛠️ 4. Herramientas Utilizadas

| Herramienta | Uso |
|---|---|
| **Bruno** | Para consumir y probar cada endpoint con distintos métodos HTTP. |
| **Swagger Editor** | Para redactar y visualizar la documentación OpenAPI 3.0. |
| **Swagger UI** | Para probar en tiempo real las rutas documentadas. |
| **GitHub** | Para almacenar y versionar la documentación del examen. |

## 🔍 5. Proceso de Documentación

1. **Creación del archivo base Swagger (openapi.yaml)**

Se definió la estructura general de la API incluyendo:

- openapi versión 3.0.0
- Información general (info)
- Servidores (servers)
- Rutas (paths)
- Componentes de esquemas y parámetros reutilizables (components)

2. **Definición de paths**

Cada endpoint fue redactado con:

- Descripción breve (summary)
- Parámetros de entrada (path o query)
- Ejemplos de requestBody (para POST/PUT/PATCH)
- Esquema de respuesta esperado

Esto permite tanto entender el uso de cada ruta como probarla desde la interfaz interactiva.

3. **Esquemas de datos (components/schemas)**

Se modelaron los objetos JSON que la API maneja (Post, Comment, Album, Photo, Todo, User), con sus atributos y tipos de datos asociados.

4. **Visualización en Swagger Editor**

Una vez redactado el archivo, se pegó en https://editor.swagger.io/ para:

- Verificación sintáctica de OpenAPI
- Interacción con los endpoints desde Swagger UI
- Validación de parámetros, request bodies y responses

## 📊 6. Resultados

La documentación generada permite:

- Visualizar todos los endpoints en una interfaz amigable.
- Probar cada ruta con parámetros y cuerpos de ejemplo directamente desde el navegador.
- Entender claramente los métodos HTTP utilizados en la práctica de Bruno.
- Consultar definiciones de cada recurso y sus esquemas asociados.

Este enfoque de documentación hace que el uso de una API REST sea accesible incluso para desarrolladores sin conocimiento previo del backend.

## 📌 8. Conclusión

Este examen permitió aplicar conocimientos prácticos y teóricos sobre:

- Métodos HTTP
- Consumo de APIs
- Documentación de APIs usando Swagger
- Uso de herramientas colaborativas como GitHub y Swagger Editor

La documentación generada mejora la comunicación entre frontend y backend, facilita pruebas automatizadas y genera un artefacto reutilizable para futuros desarrollos.