Swagger 2.0 y **OpenAPI 3.0** son especificaciones que describen servicios web RESTful. **OpenAPI 3.0** es una evolución de **Swagger 2.0**, ya que Swagger fue renombrado a OpenAPI cuando la especificación fue donada a la OpenAPI Initiative. A continuación, te detallo las diferencias principales y cuál es «mejor» según tus necesidades:

Diferencias principales:

1. Estructura del documento

- **Swagger 2.0:**
  - El documento se estructura en JSON o YAML.
  - Tiene una sección principal llamada `paths` para definir los endpoints.
- **OpenAPI 3.0:**
  - Se reorganizó la estructura del documento para ser más clara y flexible.
  - Introduce nuevas secciones como `components` para centralizar definiciones reutilizables (schemas, parameters, responses, etc.).

2. Soporte para JSON Schema

- **Swagger 2.0:**
  - Usa una versión más antigua de JSON Schema, lo que limita algunas características avanzadas.
- **OpenAPI 3.0:**
  - Integra mejor el soporte para JSON Schema y permite una definición más robusta de datos.

3. Definición de parámetros y respuestas

- **Swagger 2.0:**
  - Limitado en la definición de parámetros de diferentes tipos.
  - Las respuestas y sus contenidos son menos flexibles.
- **OpenAPI 3.0:**
  - Introduce `content` en las respuestas y en los parámetros del cuerpo (`requestBody`), permitiendo especificar múltiples tipos de medios (JSON, XML, etc.) con mayor detalle.

4. Soporte para multipart/form-data

- **Swagger 2.0:**
  - No soporta bien `multipart/form-data`, lo que complica el manejo de archivos.
- **OpenAPI 3.0:**
  - Mejora el soporte para `multipart/form-data`, facilitando la carga de archivos

y datos en la misma solicitud.

5. Referencias y reutilización

- **Swagger 2.0:**
  - Permite referencias (`$ref`) pero no es tan estructurado.
- **OpenAPI 3.0:**
  - Introduce `components`, que centraliza y organiza mejor elementos reutilizables como esquemas, parámetros, respuestas y encabezados.

6. Callbacks

- **Swagger 2.0:**
  - No soporta callbacks.
- **OpenAPI 3.0:**
  - Incluye soporte para callbacks, útil para especificar flujos asincrónicos.

7. Polimorfismo y anyOf/oneOf

- **Swagger 2.0:**
  - Soporte limitado para polimorfismo.
- **OpenAPI 3.0:**
  - Agrega soporte para `anyOf`, `oneOf` y `not`, lo que mejora la capacidad de modelar datos más complejos.

8. Documentación de seguridad

- **Swagger 2.0:**
  - Soporte básico para esquemas de seguridad.
- **OpenAPI 3.0:**
  - Permite definir múltiples esquemas de seguridad y combinarlos (OAuth, API keys, HTTP authentication).

¿Cuál es mejor?

- **OpenAPI 3.0** es superior en términos de flexibilidad, expresividad y capacidades técnicas. Es la elección recomendada para nuevos proyectos o cuando necesitas modelar APIs modernas y más complejas.
- **Swagger 2.0** sigue siendo útil si estás trabajando con proyectos heredados que ya usan esta versión y no requieren las características avanzadas de OpenAPI 3.0.

¿Cuándo usar cada uno?

- Usa **Swagger 2.0** si:
  - Tu equipo ya tiene herramientas y flujos configurados para esta versión.
  - Trabajas con APIs más simples o sistemas antiguos.
- Usa **OpenAPI 3.0** si:
  - Estás desarrollando una nueva API.
  - Quieres aprovechar las mejoras en reutilización, polimorfismo y definición avanzada.
  - Necesitas integrar medios de comunicación más complejos (archivos, tipos de respuesta variados).

En general, migrar a OpenAPI 3.0 siempre será beneficioso a largo plazo.
