Análisis de caso

---

api rest

Ashley Rodriguez.

Situación inicial
Una empresa de tecnología llamada TechTareas está desarrollando una aplicación web para la gestión interna de tareas de sus empleados. El equipo de desarrollo ha decidido implementar una arquitectura basada en microservicios, y uno de los componentes clave será la API RESTful encargada de manejar el registro, visualización, actualización y eliminación de tareas.
El producto será consumido por una app web y una app móvil. Los stakeholders piden que sea simple, bien documentada, y que tenga en cuenta la escalabilidad y el mantenimiento a futuro.

1.Define el recurso principal de la API (atributos de una "tarea").

Recurso principal: Task (Tarea)
Atributos (modelo lógico):

- id (UUID): Identificador único de la tarea.
- title (string, 1–140): Título corto y descriptivo (obligatorio).
- description (string, 0–10.000): Detalle ampliado (opcional).
- status (enum): pending | in_progress | completed | archived.
- priority (enum): low | medium | high | critical.
- due_date (string, ISO-8601, ej. 2025-08-26T15:00:00Z): Fecha límite
- reporter_id (UUID): Usuario creador/reportador.
- attachments (array): { id: UUID, filename: string, url: string }.
- created_at (ISO-8601): Fecha de creación (servidor).
- updated_at (ISO-8601): Fecha de última actualización (servidor).
- deleted (boolean)

2. Lista al menos 4 endpoints que representen operaciones CRUD, indicando qué método HTTP usarías y qué respuesta esperarías.

3. Establece un esquema de versionado y formato de datos.

- Base URL: https://api.techtareas.com/v1/
- Formato: application/json; charset=utf-8 (entrada y salida).
- Errores: application/problem+json (RFC 7807) recomendado.
- Estrategia de versionado: versión mayor en la URI (/v1, /v2) para cambios incompatibles; versiones menores/patch documentadas en el esquema OpenAPI (no rompen compatibilidad).
- Headers útiles: Deprecation, Sunset, Link (doc), ETag/If-Match (concurrencia), X-Request-Id (trazabilidad), Cache-Control.

4. Propone un ejemplo de cómo manejarías los errores comunes.

- 400 Bad Request: Parámetros/JSON mal formados.
- 401 Unauthorized: Token ausente/ inválido.
- 403 Forbidden: Sin permisos (p. ej., force=true en DELETE sin rol admin).
- 404 Not Found: id inexistente o archivado sin permiso.
- 409 Conflict: Colisión de actualización (o constraint de unicidad)

.

5. Indica cómo documentarías y probarías esta API.

Documentación:

- Definir contrato en **OpenAPI 3.1**
- Publicar Swagger UI / ReDoc en /docs y descargar colección Postman.

Pruebas:

- **Unitarias**
- **Integración**
- **Seguridad**
- **Observabilidad**

Responde:

● ¿Por qué es mejor usar métodos HTTP estándar en lugar de usar nombres de acciones en la URI?

Porque los métodos (GET, POST, PATCH, DELETE, PUT) ya expresan la intención (leer, crear, modificar, eliminar). Mantener URIs como sustantivos (/tasks) mejora la semántica,Evita endpoints verbosos tipo /createTask o /completeTask, que rompen convenciones, dificultan documentación y aumentan el acoplamiento.

● ¿Por qué JSON es el formato más recomendable?
es ligero, legible, ampliamente soportado por navegadores, serializa bien estructuras complejas y tiene excelente compatibilidad con OpenAPI y herramientas de pruebas.

● ¿Qué ventaja te da versionar una API?
Permite evolucionar sin romper clientes existentes: introducir cambios incompatibles en /v2 mientras v1 sigue operativa durante un período de deprecación.