Para manejar la autenticación y autorización en la API, optaría por un sistema basado en JSON Web Tokens (JWT), ya que tengo experiencia previa trabajando con esta tecnología en backend utilizando Spring Boot, en proyectos anteriores, he implementado JWT para proteger endpoints, asegurando que solo los usuarios autenticados puedan acceder a recursos sensibles. Esta experiencia me ha permitido comprender a fondo cómo generar, verificar y manejar tokens de manera segura. Aplicaría este conocimiento en Python, asegurándome de que los tokens incluyan claims adecuados, como roles o permisos específicos, para gestionar tanto la autenticación como la autorización. Esto permitiría establecer niveles de acceso según el perfil del usuario, como "administrador" o "usuario estándar", asegurando una gestión robusta de los permisos.

En términos de escalabilidad, mi experiencia con aplicaciones basadas en microservicios en Spring Boot me ha enseñado la importancia de diseñar aplicaciones que puedan crecer de forma horizontal. Usaría Docker para empaquetar la API en contenedores, como lo he hecho en otros proyectos, y Kubernetes para gestionar el escalado automático de instancias según la demanda.

Para endpoints que devuelven listas, implementaría la paginación de manera similar a lo que he hecho anteriormente en Spring Boot, esto incluiría permitir que los usuarios especifiquen parámetros como "página" y "tamaño" en las consultas, devolviendo respuestas optimizadas que no sobrecarguen al servidor. También agregaría metadatos útiles, como el total de registros y el número de páginas disponibles, brindando una experiencia más completa y manejable para los clientes que consumen la API.

La seguridad es un área que siempre priorizo, y mi experiencia con Spring Boot me ha preparado para implementar buenas prácticas también en Python. En otros proyectos, he utilizado ORMs como Hibernate para proteger contra inyecciones SQL, y aquí aprovecharía SQLAlchemy para aplicar el mismo principio. Asimismo, incluiría medidas contra ataques de XSS, asegurando que las respuestas estén debidamente sanitizadas, y políticas de seguridad estrictas configuradas a nivel de cabeceras HTTP. Para proteger contra ataques CSRF, implementaría tokens específicos para formularios sensibles, además, validaría todas las entradas y salidas utilizando esquemas bien definidos, y cifraría las comunicaciones con HTTPS.