API, REST, RESTful Y Las Buenas Prácticas De Desarrollo de API 's

Ricardo Alejandro Galindo Yani

CENTRO EDUCATIVO TECNICO LABORAL KINAL

Taller III

Profesor: Josué David Enrique Noj Aguilar

INDICE

INTRODUCCION

Hoy en día vivimos en un mundo totalmente conectado. Cada vez que nosotros utilizamos una aplicación en nuestro celular, hacemos más de alguna una compra en línea o publicamos algo en redes sociales, siempre detrás de todo eso hay sistemas que tienen que hablar entre sí para que todo funcione de una manera rápida y sin errores, existen las APIs, y estas funcionan como puentes que nos permiten que dos programas diferentes puedan intercambiar información de forma sencilla, sin que nos tengamos que complicar demasiado. Las APIs, o Interfaces de Programación de Aplicaciones, son herramientas muy clave que nos hacen posible que distintos servicios trabajen juntos. Gracias a estas nosotros podemos pedir comida desde una aplicación, también nos da la posibilidad de pagar con tarjeta de crédito o recibir notificaciones en nuestro dispositivo celular. Pero no solo existen las APIs, también hay algunas maneras específicas de construirlas y de las más famosas y utilizadas se llama REST. Esta es una transferencia de estado representacional es una forma bastante popular y sencilla para poder crear APIs que podamos utilizarlas por diferentes aplicaciones, sino que todo se vuelva un desastre de información. En esta investigación vamos a conocer mejor qué son las APIs, qué significa eso de REST y qué diferencia hay cuando se habla de APIs RESTful. Además, en este documento me enfocare por qué es tan importante seguir ciertas reglas o buenas prácticas al momento de construir una API, para que no solo funcione bien, sino que también sea fácil de usar, segura y pueda mantenerse en el tiempo.

API: Concepto y Aplicaciones

Una API, la cual significa ´Interfaz de Programación de Aplicaciones´, es un conjunto de reglas que nos dan la oportunidad de que diferentes programas hablen entre ellos. Básicamente este es como un traductor o un puente que conecta a dos aplicaciones para que estas puedan trabajar juntas sin necesidad de que una sepa cómo está hecha la otra. Gracias a las APIs gran parte de los sistemas pueden intercambiar datos de manera rápida, segura y ordenada, sin necesidad de complicarse demasiado.

Un buen ejemplo es cuando abrimos la aplicación del clima de nuestro celular y podemos ver que nos muestra la temperatura actual del lugar donde nos encontremos, esa información es probable que no esté siendo almacenada en la app sino que viene de otro servicio meteorológico a través de una API. Otro caso muy común es cuando en una tienda en línea como Amazon o eBay podemos pagar usando PayPal o tarjeta de crédito, lo que pasa en el fondo es que la tienda se conecta con los sistemas de pago usando APIs para que todo sea seguro y funcione bien.

Estas no solo son importantes para las apps que usamos a diario. También son súper esenciales en el mundo empresarial, porque estas se enfocan en ayudar a que las podamos integrar en plataformas enormes, como bancos, hospitales, compañías de vuelos, etc. Según IBM (2025), las APIs también permiten que los negocios puedan crear nuevos productos de una forma rápida, también que innoven de buena manera con una mayor facilidad y ofrezcan mejores servicios a sus usuarios.

REST: Principios y Características

REST, estas siglas significan ¨Representational State Transfer¨, este es un estilo de diseñar cómo las APIs deben comportarse para que todo sea más sencillo, ordenado y eficiente. Ester fue creado por Roy Fielding en el año 2000.

Entre los principios que son más importantes están:

- **Interfaz uniforme:** Todas las APIs REST utilizan los métodos estándar de HTTP, como GET (para obtener datos), POST (para enviar datos nuevos), PUT (para actualizar datos) y DELETE (para eliminar datos).
- **Comunicación sin estado:** Cada vez que el cliente le pide algo al servidor, debe enviar toda la información que sea necesaria. El servidor no guarda memoria de las peticiones anteriores, lo cual hace el sistema más simple y escalable.
- **Capacidad de almacenamiento en caché:** Las respuestas pueden ser guardadas temporalmente (cache), lo que ayuda a que todo cargue más rápido y se reduzca la saturación.
- **Sistema en capas:** Una API REST puede tener varias capas, pero para la vista del cliente todo parece ser una sola.

Todo este conjunto nos permite construir sistemas que no solo funcionan bien hoy, sino que además pueden crecer y adaptarse a futuro sin complicaciones enormes (Microsoft Learn, 2025).

RESTful: Diferencias con REST

Ahora, cuando nosotros escuchamos el término "RESTful", nos estamos refiriendo a una API que sigue de manera correcta y estricta todos los principios del REST. No basta con usar simplemente HTTP para decir que una API es RESTful.

Una API realmente RESTful tiene que cumplir con varias reglas y estas pueden ser como usar bien los métodos HTTP, manejar los recursos de forma organizada utilizando URLs claras, también permitir navegación a través de los enlaces que proporciona en sus respuestas, entre otras cosas (API7.ai, 2023).

Un buen ejemplo es si tenemos una API que nos permita gestionar usuarios, debería seguir rutas como “/usuarios” para obtener una lista de usuarios utilizando el método Get, también podemos utilizar “/usuarios/:id” para ver el detalle de un usuario específico, o /usuarios con POST para que podamos crear un usuario nuevo. Si en lugar de eso usamos rutas raras como /crearUsuario o /hacerListaUsuarios, entonces aunque nosotros estemos usando los métodos HTTP, realmente no estaríamos siguiendo REST correctamente y por lo tanto no sería una API RESTful.

Por eso, ser "RESTful" implica muchas cosas más que simplemente utilizar los servicios de internet o enviar datos por HTTP. Esto quiere dar a entender que respetar un conjunto de reglas muy claras que ayudan a que las APIs sean mucho más organizadas, predecibles y fáciles de entender para cualquier persona que la utilice.

Buenas prácticas en el desarrollo de APIs

Crear una API no es solo hacer que funcione es también importante que sea fácil de entender, segura y que se pueda mantener con el paso del tiempo. Para lograrlo, existen varias buenas prácticas que recomiendan los expertos, como IBM (2025) y Microsoft (2025)

- **Versionado de la API:** A medida que la API evoluciona, es importante crear versiones, un ejemplo puede ser "/api/v1/usuarios", para no romper el funcionamiento de los clientes que utilizan versiones anteriores.
- **Uso adecuado de códigos de estado HTTP:** No basta con devolver solo un estado "200" para todo. Se deben usar códigos adecuados como el 201 cuando se crea un nuevo recurso, 400 si hubo un error en la solicitud, 404 si no se encontró lo que se buscaba, 500 si hubo un error interno, entre otros.
- **Seguridad:** La autenticación y autorización son muy importantes. Se deben usar métodos como Auth o tokens JWT para asegurarse que solo las personas que estén autorizadas puedan acceder o modificar los datos.
- **Documentación clara y accesible:** Tener una buena documentación que explique cómo usar la API, qué rutas existen, qué datos se deben enviar y qué respuestas se pueden esperar es básico. Y las herramientas como Swagger u OpenAPI nos pueden ayudar a hacer documentaciones que nos facilitan mucho el trabajo.

- **Consistencia en las URLs:** Usar los nombres en plural, sin verbos, y de manera consistente ya que es mejor utilizar "/productos" que "/obtenerProducto", para que todo se vea de manera ordenada y profesional.
- **Manejo adecuado de errores:** Si algo sale mal la API debe decir claramente qué pasó y cómo solucionarlo, no simplemente devolver errores de una manera genérica.
- **Paginación y filtrado:** Cuando la API maneja muchos datos, es importante ofrecer paginación un ejemplo podría ser, devolver solo 10 usuarios a la vez, y permitir filtros como buscar solo usuarios de una ciudad para hacer todo más rápido y menos pesado.

Además de todo esto, también es importante pensar en la experiencia del usuario que va a consumir nuestra API. Ya que no basta con que técnicamente funcione, debe ser intuitiva, predecible y fácil de integrar en otros proyectos. Un ejemplo que ejemplifica esto, es si alguien tarda mucho tiempo en entender cómo hacer una simple consulta o si las respuestas que devuelve la API no son muy claras, es probable que la persona que la está utilizando busque otra solución. Una buena API debe ser como un buen restaurante, ya que no solo debe servir comida, sino que debe de ofrecer un servicio que haga que la gente quiera volver. Por eso cuidar pequeños detalles como tiempos de respuesta rápidos, mensajes amigables de error y ejemplos claros en la documentación hacen una diferencia enorme cuando se trata de construir APIs que realmente marque la diferencia.

CONCLUSIONES

Con el tiempo las API han cambiado radicalmente la forma en que las aplicaciones interactúan, y ha permitido a los sistemas que sean completamente distintos puedan comunicarse entre sí de una manera rápida, ordenada y eficiente. Gracias a estas el día de hoy podemos pedir comida desde aplicaciones, poder transferir dinero en línea, recibir notificaciones en nuestro dispositivo móvil, entre otros. Todo esto sucede sin que el usuario tenga que preocuparse de la complejidad que existe detrás, ya que las APIs actúan como puentes invisibles que hacen posible esta comunicación. Con muchas formas de diseñar APIs, REST ha logrado en destacar como uno de los estilos arquitectónicos más populares y efectivos ya que tiene como objetivo principal porque propone reglas simples que buscan mantener las cosas ordenadas, fáciles de usar y escalables, lo cual es crucial cuando se manejan millones de interacciones diarias. Aplicar correctamente los principios de REST no es solo seguir un conjunto de reglas técnicas, sino pensar en el futuro del proyecto, en su crecimiento y en su facilidad de mantenimiento. Diseñar implica prestar atención a detalles como el versionado, que este es esencial ya que ayuda a no romper sistemas viejos cuando salen nuevas actualizaciones, la seguridad, que protege tanto los datos del usuario como la integridad del sistema, y una buena documentación, que facilita enormemente la vida a los desarrolladores que trabajarán con la API. Incluso los aspectos más pequeños como mantener una consistencia en los nombres de las rutas o manejar los errores de una manera que sea clara pueden hacer la diferencia entre una API exitosa que todo el mundo quiera usar y no una que termina siendo abandonada con el paso de los años.

RECOMENDACIONES

Mi recomendación es que, cuando vayamos a crear una API, no solo nos enfoquemos en que funcione y ya, sino que busquemos hacer algo que sea claro, seguro, fácil de mantener y obviamente que sea útil para todos los que la usen a diario. Es importante que sigamos aprendiendo todo el tiempo, porque el mundo de las APIs cambia muy rápido siempre salen nuevas tecnologías, mejores prácticas y herramientas todo el tiempo. También es muy importante hacer revisiones y pruebas constantes, no solo al principio cuando digamos "todo se ve bonito" sino durante todo el ciclo de vida de la API, esto hacerlo más que todo para asegurarnos de que sigue funcionando bien en diferentes situaciones reales. Además es importante escuchar a los usuarios y tomar en serio su feedback que nos puede ayudar a descubrir fallos que a simple vista no podemos ver, mejorar funciones y hacer que la experiencia general sea muchísimo mejor, además es recomendable usar formatos claros como JSON que hace una gran diferencia, porque así otros desarrolladores entienden más rápido cómo poder integrar nuestra API y evitan errores innecesarios. Es clave que utilicemos rutas claras, respuestas fáciles de entender, mensajes de error específicas y una buena documentación que sea accesible para marcar gran diferencia entre una API buena y una que nadie quiere usar, ottro punto que no podemos ignorar es la seguridad siempre debemos proteger los datos, validar correctamente la información que entra y controlar quién accede, es obligatorio desde el primer día y por último, siempre hay que pensar en el futuro de una API, si está bien pensada desde el principio va a ser mucho más fácil de escalar, de agregar nuevas funciones, y de sobrevivir a los cambios tecnológicos sin convertirse en un problema que se vuelva incontrolable.

REFERENCIAS

1 Bibliografía

API7.ai. (2023). *APIs RESTful: Principios y Mejores Prácticas*. Obtenido de https://api7.ai/es/learning-center/api-101/restful-api-best-practices

Bytes, C. (2022). *Buenas prácticas y diseño de una API REST*. Obtenido de https://coffeebytes.dev/es/buenas-practicas-y-diseno-de-una-api-rest/

IBM. (2025). *¿Qué es una API REST?* Obtenido de https://www.ibm.com/es-es/topics/rest-apis

Learn, M. (2025). *Diseño de API web RESTful - Azure*. Obtenido de https://learn.microsoft.com/es-es/azure/architecture/best-practices/api-design

Wikipedia. (2025). *Transferencia de Estado Representacional*. Obtenido de https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional