Estrategia y arquitectura de API: Una estrategia coordinada

ca technologies

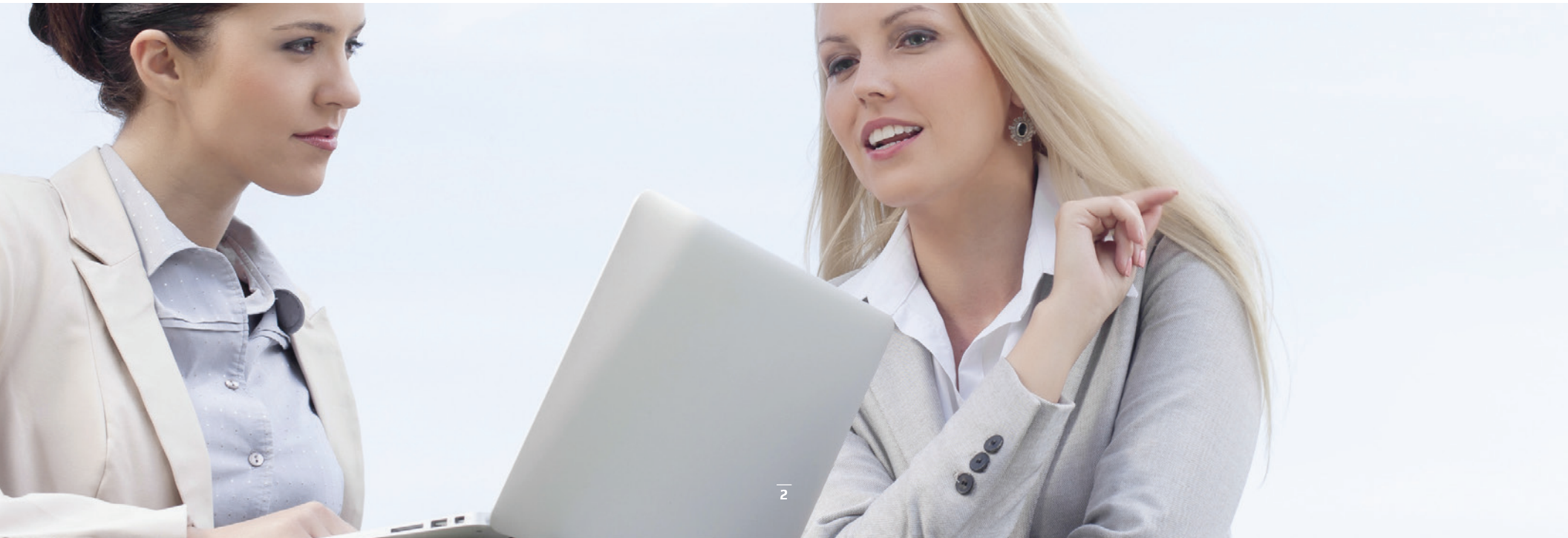
Introducción

El surgimiento de la interfaz de programación de aplicaciones (API) representa una oportunidad empresarial y un reto técnico. Para los líderes empresariales, las API representan la oportunidad de abrir nuevos flujos de ingresos y maximizar el valor del cliente. Pero los arquitectos empresariales son los que están a cargo de crear las API que hacen que los sistemas administrativos estén disponibles para volver a utilizarlos en nuevas aplicaciones web y móviles.

Es fundamental que todas las partes interesadas comprendan que los objetivos empresariales y los retos técnicos de un programa de API están íntimamente relacionados. Los administradores del programa deben hacerse responsables de comunicar con claridad los objetivos empresariales clave de una API propuesta a los arquitectos que serán los que desarrollarán la interfaz.

Mientras tanto, los arquitectos deben asumir la responsabilidad de mantener un enfoque claro sobre estos objetivos a lo largo del proceso de implementación de una infraestructura de API y de diseño de la propia interfaz. Todas las decisiones técnicas deben contribuir a la creación de una interfaz que le permita a los desarrolladores crear aplicaciones que los usuarios finales realmente valorarán.

Este libro electrónico describe las mejores prácticas para diseñar API centradas en resultados que se convertirán en la base del éxito de su programa de API.



Primera parte: De SOA a API

La TI empresarial del siglo 21 ha estado caracterizada por un movimiento hacia la apertura de bases de datos y aplicaciones que anteriormente estaban en silos, para que los datos y las funcionalidades puedan ser accesibles en los límites organizativos y volver a ser utilizados en sistemas nuevos. La manifestación inicial de esta tendencia proviene de la arquitectura orientada a los servicios (SOA) y la más reciente ha sido la explosión de las API orientadas a la Web.

En un nivel, los "servicios web" centrales a SOA representan lo mismo que las API web. Ambas son interfaces utilizadas para abrir sistemas administrativos. Sin embargo, hay algunas diferencias fundamentales entre las dos tecnologías, que son muy importantes para tomar decisiones básicas de diseño:

- La principal diferencia técnica es que los programas de SOA están centrados en la creación de servicios web para facilitar integraciones internas y de servidor a servidor, mientras que las API web existen para acelerar la creación de aplicaciones web y móviles, a menudo de una naturaleza centrada en el cliente.
- Los programas de SOA están impulsados generalmente por los departamentos de TI y centrados en el ahorro de costos, mientras que los programas de API se originan más comúnmente con organizaciones de desarrollo empresarial y se centran en la generación de nuevos ingresos.
- La mayoría de los proyectos de SOA están creados por y para arquitectos empresariales con el fin de ayudarlos a integrar de manera más simple los sistemas heterogéneos y entregar nuevos servicios de TI. Los programas de API, por contraste, se deben centrar en satisfacer las necesidades de los desarrolladores de aplicaciones.

Figura 1: SOA en comparación con API

	SOA	API
Objetivo de integración	Interno o para socios de negocios	Externo, a veces para socios de negocios
$$$ **Impulsor del proyecto**	Costos de TI	Ingresos empresariales
Consumidor de la interfaz	Arquitectos empresariales	Desarrolladores de aplicaciones

Objetivos del diseño de la API

Sin embargo, muchos programas de API están creciendo sin control con respecto a iniciativas anteriores de SOA. Los servicios web centrados en integraciones internas o de socios de negocios se están abriendo para los desarrolladores; tanto dentro como fuera de la empresa. Durante este proceso, es importante que los diseñadores de API recuerden que un programa de API tiene impulsores y requisitos diferentes a los que inicialmente guiaban a las empresas para que abran sus activos de TI a través de servicios web.

Con esto en mente, los objetivos generales del diseño de API se pueden definir de la siguiente manera:

- Permitir el autoservicio para desarrolladores de aplicaciones y usuarios de aplicaciones semejantes.
- Reducir barreras para acceder a recursos empresariales valiosos.
- Priorizar las necesidades y preferencias de desarrolladores de aplicaciones cliente.
- Fomentar la colaboración entre recursos internos y externos.
- Abordar los problemas de seguridad y escalabilidad al exponer activos de TI al mercado abierto.

Ante todo, el diseño de API debe estar centrado en maximizar el valor empresarial de la interfaz. En la segunda parte, le echaremos un vistazo de cerca a cómo las API agregan valor a la empresa.

Segunda parte: La cadena de valores de API

Es posible que las API no tengan un valor inherente, pero aportan un enorme valor a la empresa. Lo logran a través de sus datos administrativos y la funcionalidad de la aplicación que la interfaz habilita. En esta vista, la API es solamente un facilitador que permite que los sistemas con un gran valor organizativo sean reutilizados en aplicaciones que tienen más probabilidades de brindar un valor empresarial directo.

Aunque es una perspectiva útil, cuando se mira de cerca, se vuelve más claro que una API bien diseñada es, de hecho, un conector potente y complejo. Une una amplia variedad de activos empresariales (sistemas de TI, personal interno y externo, aplicaciones cliente y consumidores) para comprender de manera más eficaz el valor potencial de estos activos. Podemos referirnos a esta situación como “la cadena de valores de API”.

Es importante comprender que una API brinda valor de esta manera relativamente compleja ya que, en caso contrario, es demasiado fácil perder la vista del hecho de que las API existen para brindar valor empresarial, no eficiencias técnicas. Pero aunque las API brindan valor de manera más directa que SOA, lo hacen de manera menos directa que la Web basada en navegadores, donde un sitio puede brindar ventas potenciales o reales. Las API generan ingresos de manera más sutil al vincular los diversos activos descritos a continuación.

Figura 2: La cadena de valores de API

Algunos ejemplos de cómo las API generan valor

Cualquier API tendrá su propio valor único. En términos generales, las empresas pueden usar una API como una forma para:

Generar nuevos ingresos de manera directa

Una API puede ser una fuente directa de ingresos si los desarrolladores deben pagar para acceder o si la interfaz se utiliza para facilitar la creación dentro de la empresa de aplicaciones pagas o para habilitar el comercio electrónico.

Extender el alcance y el valor del cliente

Las API simplifican el proceso de comunicación con nuevos clientes o de aumento del valor de clientes actuales al ofrecer servicios existentes a través de plataformas y dispositivos nuevos.

Respaldar actividades de marketing y ventas

Una API también puede ayudar a una empresa a comercializar sus productos y servicios al permitir la creación del tipo de funcionalidad atractiva y envolvente asociada con las mejores prácticas de marketing en línea.

Estimular la innovación empresarial y técnica

Las API ayudan a las organizaciones a desarrollar sistemas, ofertas y estrategias nuevas ya que reducen las barreras de la innovación al posibilitar la implementación de ideas sin cambiar los sistemas administrativos.

Tomar decisiones sobre el diseño

Las decisiones sobre el diseño de la API deben basarse en qué vinculará la API precisamente; qué habrá de ambos lados de la interfaz, dentro de la infraestructura organizativa de TI y fuera del firewall de la empresa. Específicamente, es fundamental responder estas dos preguntas:

- ¿Cuáles sistemas están siendo expuestos y dónde (y con quién) residen?
- ¿Quiénes son los desarrolladores objetivo y qué tipo de aplicaciones desarrollan?

"¿Quiénes son los desarrolladores objetivo?" es una pregunta particularmente importante y relevante para la manera más fundamental en la que se categorizan las API; como "privada" o "abierta". Las API privadas solo pueden utilizarse dentro de la empresa o, en algunos casos, por organizaciones de socios de negocios. Las API abiertas están disponibles para una comunidad más amplia de desarrolladores externos, que son libres para crear sus propias aplicaciones a través de los recursos administrativos de la empresa.

Las API privadas se asemejan más a los servicios web. Normalmente, el objetivo de una API privada será ayudar a los desarrolladores, contratistas o socios de negocios internos a crear de manera más eficaz aplicaciones para el uso interno o externo. Como sucede con los servicios web, los ahorros de costo representan el impulsor clave ya que las API permiten el desarrollo de nuevas aplicaciones de una manera rentable. Sin embargo, se utilizan muchas API privadas para crear aplicaciones móviles y web centradas en el público que generan nuevo valor empresarial de manera más eficaz.

Los programas de API abierta tienden a centrarse en la adopción. Al permitirle a desarrolladores de terceros acceder a sus API, las empresas buscan que sus activos de TI estén disponibles para la base de usuarios más amplia posible. Por lo tanto, la adopción de desarrolladores es una métrica clave para medir el éxito de una API abierta. Aunque hay menos API abiertas que API privadas, es en las API abiertas donde residen las mayores oportunidades empresariales y los riesgos técnicos/retos de diseño más importantes.

De hecho, las API abiertas no solo crean una gama completamente nueva de retos del diseño de integración (por ejemplo, cómo abrir sistemas administrativos para los desarrolladores externos sin exponer esos sistemas a hackers), también pueden crear nuevos riesgos empresariales. Un programa de API abierta con conceptos escasos puede llevar a una empresa a desplazar su propio negocio principal y exponer potencialmente los activos comerciales críticos de la empresa a competidores.

Las consideraciones empresariales como estas deben impulsar la toma de decisiones sobre el diseño técnico. Debatiremos sobre cómo alinear las consideraciones empresariales con las decisiones técnicas más adelante en la tercera parte.

Tercera parte: Alineación del diseño de API con los objetivos empresariales

Mientras SOA ha buscado históricamente mejorar los procesos organizativos, los programas de API buscan aumentar los ingresos empresariales. Por lo tanto, las decisiones sobre el diseño de la API deben centrarse en los principales objetivos empresariales y estratégicos del programa de API de la empresa. Antes de comenzar a diseñar una API, debe ser claro con respecto a los problemas que el programa de API busca resolver, las oportunidades que busca conseguir y la manera en que lo logrará. Específicamente, es importante responder estas preguntas:

- ¿Qué activos estarán disponibles?
- ¿Cómo debe hacer la API para que estos activos estén disponibles?
- ¿Qué tipo de aplicaciones se pueden desarrollar con la API?
- ¿De qué manera los desarrolladores pueden estar motivados para utilizar la API?
- ¿De qué manera las aplicaciones crearán valor para la empresa?

La comunicación y la colaboración son las claves para diseñar una API que aborde estos retos y estas oportunidades. A lo largo del proceso de diseño, implementación y administración de una interfaz, los administradores del programa y los arquitectos de API deben trabajar estrechamente para asegurarse de estar de acuerdo con sus objetivos estratégicos principales, qué harán para lograr esos objetivos y cómo evaluarán los resultados de sus esfuerzos. Específicamente, los roles empresariales y técnicos deben estar de acuerdo en:

- El objetivo y el estado final ideal del programa.
- Las tareas iniciales que permitirán que la organización trabaje para lograr esos objetivos.
- Las métricas clave que se utilizarán para medir el éxito.
- Las tareas diarias que permitirán que el programa continúe cumpliendo sus objetivos.

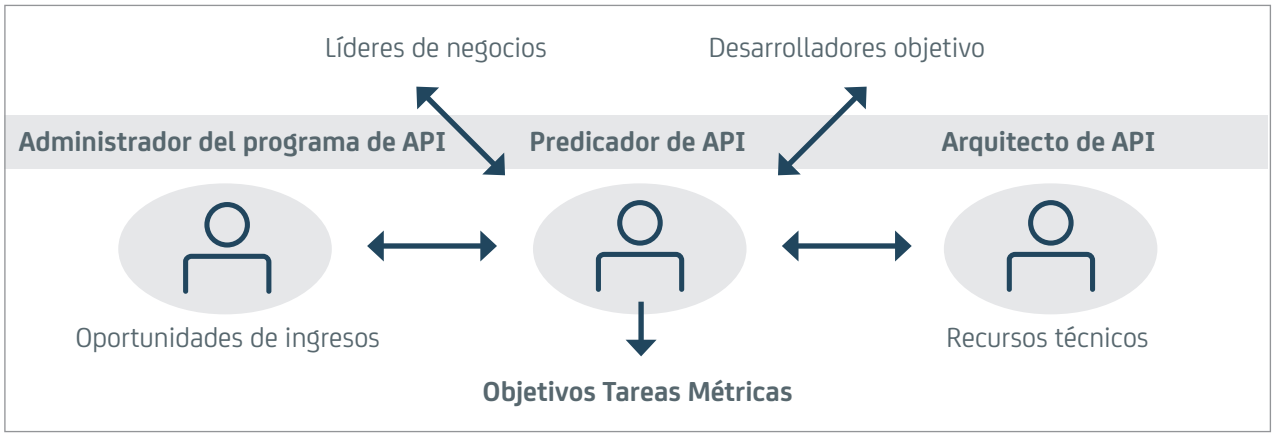
Asignación de un patrocinador

Para asegurarse de que los administradores y arquitectos empresariales estén en la misma página, el programa debe tener un "patrocinador" que sea capaz de abarcar la división que a veces aparece entre los departamentos técnicos, administradores empresariales y desarrolladores de aplicaciones. A menudo, las organizaciones cometen el error de asignar este rol a un administrador de marketing no técnico, pero este "predicador de API" debe poder comprender las restricciones arquitectónicas de la organización y compartir los intereses de los desarrolladores de aplicaciones.

El rol del predicador es establecer una comunicación clara con todos los interesados, específicamente:

- "Vender" el programa de API a los ejecutivos y otras personas sénior que toman decisiones.
- Asegurarse de que los arquitectos de API comprendan los objetivos empresariales de los administradores del programa.
- Ayudar a los administradores del programa a comprender las restricciones y los recursos técnicos de los arquitectos.
- Recopilar información sobre las preferencias y los requisitos de los desarrolladores.

Figura 3: Alineación de los objetivos de API



Una vez que se establece la comunicación y se acuerdan los objetivos, las tareas y las métricas, el verdadero trabajo del diseño de API puede comenzar. Hablaremos de este trabajo en la cuarta parte.

Algunas notas sobre la estrategia empresarial de API

Los administradores del programa (o “propietarios de API”), junto con el predicador de API de la organización, tienen que asumir la responsabilidad de crear una estrategia empresarial de API clara y comunicar esta estrategia a las personas de nivel ejecutivo que toman las decisiones, como también a los arquitectos y desarrolladores que implementarán la parte técnica de la estrategia.

El primero paso es establecer un objetivo empresarial claro y una visión para el programa de API que esté alineada con la visión más amplia de la empresa. Una API no es meramente una solución técnica y debe ser tratada como un producto o una estrategia empresarial por sí misma; aunque integrada dentro de la estrategia empresarial general.

Con esto en mente, el próximo paso debe ser desarrollar un modelo empresarial alrededor de esta visión, describiendo los detalles de:

Costos, recursos y eficiencias

- Los sistemas, las relaciones, las actividades y otros recursos que el programa aprovechará y cómo el programa permitirá que la empresa utilice de mejor manera estos recursos.

Valor, ingresos e innovación

- Los clientes, mercados y canales a los que el programa apuntará y cómo la innovación técnica posibilitará la generación de nuevos ingresos a partir de estos objetivos.

El núcleo de este modelo empresarial debe ser una propuesta de valor que describa claramente el valor empresarial, mensurable y real que el programa de API le ofrecerá a la empresa.

Cuarta parte: Diseño de una API utilizable

Desde una perspectiva completamente técnica, diseñar una API es relativamente fácil. Sin embargo, diseñar una que aporte valor real a la empresa puede complicar la situación. Además de la funcionalidad, los arquitectos también deben tener en cuenta los objetivos empresariales y la experiencia del usuario final.

Esto puede representar un reto especialmente para cualquiera que esté ampliando un proyecto de SOA en un campo de API. En SOA, las necesidades del arquitecto son centrales y se asume la adopción del usuario. Consecuentemente, los arquitectos con formaciones de SOA normalmente abordarán las decisiones de diseño de API con el supuesto de que los usuarios de la aplicación y la interfaz tendrán las mismas necesidades y predisposiciones que ellos. Esto casi siempre lleva a tomar malas decisiones sobre el diseño.

Con las API, el enfoque del diseño no debe estar en la funcionalidad, sino en la experiencia del usuario. La pregunta clave no es "¿Qué funcionalidades necesito exponer?" sino "¿Cómo utilizarán los desarrolladores esta interfaz?" Si los desarrolladores no desean utilizar su API, entonces no tiene valor. Por lo tanto, el diseño debe estar centrado en el desarrollador y enfocado en proporcionar la barrera más baja posible de entrada para la audiencia de desarrolladores objetivo.

Ya sea si una API se publica de manera privada o abierta, una buena experiencia del desarrollador (DX) es fundamental para que tenga éxito. La DX es mucho más difícil de cuantificar que la funcionalidad expuesta. Aunque puede definirse como la suma de interacciones entre el proveedor y el desarrollador de la API, el resultado de esta suma no representa tanto un número sino un sentimiento: ¿de qué manera la interfaz hace que se sientan los desarrolladores?

Obviamente, es una métrica difusa pero hay pasos, sin duda prácticos, que puede tomar en el mundo real para comprender cómo es probable que sus desarrolladores se sientan acerca de las diferentes estrategias que puede tomar en el diseño de su API. Específicamente, debe:

- Crear perfiles de los desarrolladores.
- Hacer un prototipo y probar su API en el campo.

Perfiles de los desarrolladores

No puede crear una API utilizable a menos que conozca las necesidades y preferencias de su desarollador objetivo. Hay una tendencia de asumir que los desarrolladores que desarrollan aplicaciones cliente en API son autodenominados “hackers” jóvenes, obsesionados con los últimos lenguajes y protocolos. Sin embargo, en muchos casos (particularmente, en escenarios de API privada), los desarrolladores de servicios empresariales permanecen leales a maneras más arraigadas de hacer las cosas.

La cuestión es que cada proyecto de API necesitará abordar una audiencia de desarrolladores particular para poder tener éxito. En algunos casos, este puede ser un grupo muy homogéneo con necesidades compartidas. En otros casos, es posible que necesite abordar una amplia variedad de preferencias. Independientemente, debe comprender quién estará utilizando su API y cómo puede definir la interfaz para asegurarse de que esos desarrolladores puedan utilizar de manera rápida y eficaz sus recursos administrativos.

Por lo tanto, el primer paso es formar una persona (o un conjunto de personas) para definir el tipo (o tipos) de desarrolladores a los que está apuntando con sus API. Debe incluir información sobre:

- Para quién trabajan (o en qué departamento) y por qué están desarrollando una aplicación.
- Habilidades de programación, restricciones técnicas y preferencias de lenguaje/protocolo.
- Temperamento personal y en qué contexto trabajan mejor.

Creación de un prototipo

Una vez que tiene conocimientos de los objetivos del trabajo, los requisitos técnicos y las preferencias personales de sus desarrolladores objetivo, puede comenzar a desarrollar una interfaz que aborde esos criterios. Sin embargo, antes de crear una API de producción relacionada a sistemas administrativos o datos reales, debe desarrollar un prototipo ligero que se pueda cambiar con más facilidad. Este prototipo le permitirá probar los supuestos del diseño que ha realizado según sus personas objetivo.

Figura 4: Herramientas útiles para la creación de prototipos de API

Una de las ventajas de desarrollar un prototipo ligero basado en funcionalidades o datos “desechables” es que le permite aplicar seguridad mínima y proporcionar la menor barrera posible de entrada para los desarrolladores. Esto le permitirá atraer a sus desarrolladores objetivo desde un principio. Escribirán aplicaciones ligeras para probar su diseño de API y proporcionarán comentarios. Luego, puede realizar cambios en la interfaz y probarla nuevamente. Después de algunas iteraciones, debería ir por buen camino.

Por supuesto, ninguna de estas acciones se relaciona con cómo tomará decisiones reales y fundamentales acerca del diseño de la interfaz. En la quinta parte, comenzamos a debatir sobre las opciones reales del diseño de API.

Quinta parte: Estilos de API

Elegir un estilo de API es una de las decisiones más importantes que debe tomar un diseñador de interfaces. Las decisiones de este tipo se verán afectadas inevitablemente por consideraciones técnicas, como la naturaleza específica de recursos administrativos que se exponen o las restricciones de la organización de TI. Sin embargo, también se deben tener en cuenta otros aspectos como los objetivos empresariales del programa de API y las necesidades y preferencias de la audiencia de desarrolladores objetivo.

Los estilos comunes de diseño de API en la actualidad se pueden recaen en las siguientes categorías:

- Servicio web (también conocido como túneles)
- REST pragmática (también conocida como URI)
- Hipermedios (también conocido como “REST verdadera”)
- Impulsado por eventos (también conocido como IoT)

Servicio web

El estilo Servicio web es una estrategia basada en operaciones y válida para diferentes transportes para el diseño de API, que utiliza Lenguaje de descripción de servicios web (WSDL) para describir interfaces. Viene del mundo SOA, donde las interfaces de servicio web eran utilizadas para integrar redes heterogéneas. Por lo tanto, puede ser una buena opción de estilo si su programa involucra ampliar las interfaces de SOA. La gran cantidad de herramientas que existen para los servicios web también significa que las aplicaciones cliente a menudo se pueden desarrollar de manera rápida y simple.

Sin embargo, este estilo tiene limitaciones graves. Primero, aunque este estilo válido para diferentes transportes puede usar protocolo de transferencia de hipertexto (HTTP), no es muy eficiente en este contexto. Por lo tanto, no es la mejor opción si sus servicios están siendo ampliados a la Web abierta.

Además, solo resulta práctico si sus desarrolladores objetivo conocen los estándares de SOA como WSDL, Protocolo simple de acceso abierto (SOAP) y Llamada de procedimiento remoto (RPC). Para la mayoría de desarrolladores cliente, la curva de aprendizaje probablemente sea empinada.

Particularmente en escenarios de API abierta y especialmente para las personas centradas en dispositivos móviles. Como regla, a los desarrolladores de aplicaciones no les gusta SOAP como lenguaje de programación y la herramienta disponible para desarrollar tendencias de servicios web cliente no son compatibles con dispositivos móviles. Dejando de lado las consideraciones prácticas, hay un problema de percepción: a través del uso del estilo de Servicio web puede hacer que su organización parezca un “dinosaurio” lento, lo que lo llevará a disminuir la adopción entre los desarrolladores de aplicaciones móviles.

REST pragmática

El estilo Transferencia de estado representacional (REST) pragmática es una estrategia que se centra más en la Web para diseñar las interfaces de integración. Este estilo, que utiliza URI en lugar de WSDL y tiene transporte específico (es compatible exclusivamente con HTTP), ha sustituido considerablemente el estilo de Servicio web en el diseño de API empresarial. De hecho, el término “API web” se utiliza indistintamente con “API RESTful”, y lograr “RESTfulness” a menudo se considera un objetivo clave de cualquier proyecto de diseño de interfaz.

En efecto, la mayoría de las API REST que se utilizan en la actualidad no cumplen por completo los criterios de REST descritos en la tesis determinante de doctorado de Roy Fielding del 2000. Mientras REST fue definida para describir formalmente el tipo de interacciones hipervinculadas y dinámicas que potencian la Web, la mayoría de las API web abordan el intercambio de datos estáticos. Por lo tanto, a modo de argumento, es más preciso referirse a este estilo de diseño como “REST pragmática”.

Es fácil ver por qué el estilo REST pragmática se ha popularizado tanto. Debido a que el URI es intuitivo y los desarrolladores web y móviles están más familiarizados con las interfaces de RESTful, es probable que la adopción y productividad de los desarrolladores sean altas. Además, enfocarse en HTTP hace que las API REST pragmáticas sean ideales para desarrollar las aplicaciones móviles y web de la actualidad. Ahora mismo, es probable que sea el estilo predilecto para la mayoría de los proyectos.

Sin embargo, el estilo REST pragmática no es perfecto para todos los contextos y parece probable que los desarrolladores futuros desafíen su dominio. Hay diferencias definidas con este estilo: es limitado para cuatro métodos, puede tener “exceso de comunicación” y el diseño de URI no es estándar. Además, con la Internet de las cosas (IoT) y la gran base de datos que se están expandiendo y modificando considerablemente en la red en línea, es probable que haya retos para esta estrategia específicamente centrada en la Web.

Hipermedios

El estilo de diseño de API de hipermedios es una estrategia basada en tareas que tiene como objetivo proporcionar una alternativa sostenible a la REST pragmática. Como la REST pragmática, las API de hipermedios están centradas generalmente en estándares de URI, HTTP y RESTful. Pero en cierto sentido, el estilo de Hipermedios representa una aplicación más fiel de la arquitectura de RESTful, según Fielding, que describe por qué la Web ha demostrado ser tan escalable.

Como tal, la estrategia de Hipermedios está incluso más centrada en la Web: los hipervínculos y formularios de la Web están reflejados de la manera en que una API de hipermedios proporciona vínculos para navegar a través de flujos de trabajo y entradas de plantillas para solicitar información. Como la arquitectura de RESTful de la Web ha demostrado ser muy escalable y evolutiva, una API de hipermedios bien diseñada puede seguir respaldando aplicaciones nuevas durante años.

Aunque esta estrategia de arquitectura es evidentemente una opción atractiva para empresas que buscan crear API escalables que respaldan de manera confiable aplicaciones web y móviles durante un periodo largo, aún es un estilo de diseño emergente con una notable falta de herramientas asociadas. Esto puede impactar las tasas de adopción de los desarrolladores y dificultar la creación rápida de aplicaciones cliente potentes por parte de aquellos desarrolladores que adoptan la API.

Impulsado por eventos

Aunque los estilos centrados en HTTP como REST pragmática e Hipermedios pueden ser ideales para aplicaciones web y móviles como las conocemos, la llegada de HTML5 e IoT está cambiando las cosas; creando la posibilidad de aplicaciones más dinámicas, pero también demandando interfaces más ligeras. En este contexto, el estilo Impulsado por eventos ha surgido como una alternativa válida para diferentes transportes, ideal para permitirle a las aplicaciones utilizar WebSocket y otras alternativas emergentes de HTTP.

Este estilo, que se centra en el servidor o en eventos iniciados por el cliente, proporciona una opción de bajos gastos, capaz de entregar un mejor desempeño en situaciones donde hay una gran cantidad de mensajes pequeños que pasan entre el sistema administrativo y la aplicación.

Por lo tanto, es ideal para IoT y una gama de casos de uso móviles; especialmente mensajería instantánea, chat por video, juegos con múltiples jugadores, etc. También es probable que resulte atractivo para los desarrolladores más vanguardistas.

Por supuesto, no todos los desarrolladores están tan obsesionados con estar a la vanguardia y hay muchos casos de uso donde una estrategia convencional de RESTful sería más adecuada. HTTP es aún el protocolo de transporte que impulsa la Web y no adapta eventos enviados por el cliente particularmente bien. Además, el modelo de solicitud de respuesta en el que este estilo está desarrollado hace que el desarrollo de aplicaciones cliente sea más complejo para los desarrolladores.

Figura 5: Estilos de arquitectura para el diseño de API

El estilo que elija dependerá de sus restricciones técnicas, objetivos empresariales y preferencias de los desarrolladores. Asegúrese de no caer en la trampa de adoptar un estilo “de moda” si no es adecuado para su contexto específico. Al mismo tiempo, intente elegir un estilo que demostrará ser escalable y adaptable a largo plazo, mientras sus recursos cambian, su audiencia de usuarios crece y la propia naturaleza de la red en línea evoluciona.

Sin importar el estilo que elija, hay ciertos componentes de arquitectura que querrá que su API incluya.
En la sexta parte, describiremos estos componentes y cómo estarán organizados.

Sexta parte: Arquitectura de API

Los estilos del diseño de arquitectura descritos previamente deben proporcionar un modelo para la manera en que debe diseñar el marco arquitectónico que habilita la funcionalidad única de su API. Ciertos casos de uso exigirán la implementación de estilos de diseño específicos. También es importante observar que hay una cantidad de componentes que deben incluirse en cualquier arquitectura de API, sin importar el caso de uso.

Estos componentes arquitectónicos comunes no deben desarrollarse en la implementación de cualquier API. En cambio, se deben implementar en una infraestructura de API principal que estará entre las API de la organización y las aplicaciones cliente que impulsan esas API. Separar estos componentes hace más rápido y simple diseñar API adicionales, actualizar una gama de API conjuntamente y garantizar la ejecución sin inconvenientes de API, sistemas administrativos y aplicaciones cliente.

Para obtener una eficacia máxima, estos componentes deben estar organizados en capas, para que todo el tráfico de datos deba pasar a través de cada una de las capas nombradas en la derecha, en el orden específico.

Figura 6: Capas arquitectónicas

La capa de seguridad

Así como abren un mundo de oportunidades empresariales, las API tienen el potencial de abrir la empresa a nuevas amenazas de seguridad graves al exponer sistemas administrativos y datos confidenciales al mundo exterior. Las API son vulnerables a muchas de las amenazas de seguridad que han plagado la Web, además de una gama de nuevas amenazas específicas de API. Por lo tanto, es fundamental implementar seguridad sólida y específica de API en la periferia de la arquitectura de su API.

Esta necesidad de seguridad sólida puede estar en conflicto con un objetivo básico del diseño de API; una API bien diseñada facilita que los desarrolladores creen aplicaciones que proporcionen acceso sin inconvenientes a los recursos de la empresa. Es probable que la seguridad sólida tenga un impacto en esta facilidad de acceso. Implementar seguridad en una arquitectura de API centralizada (en lugar de en la implementación de API) ayudará a aliviar este impacto, como también posibilitará el uso de tecnologías de administración de acceso flexible como OAuth y OpenID Connect.

La capa de almacenamiento en caché

La eficiencia de la interfaz demostrará que es esencial para proporcionar experiencias fluidas para desarrolladores y usuarios finales necesarias para cumplir con los objetivos de adopción y retención del programa de API. Una manera de maximizar la eficiencia de API es colocar una capa de almacenamiento en caché cerca de la periferia de la arquitectura de API. Esta capa debe permitir la entrega de respuestas en caché para solicitudes comunes, lo que reduce la presión colocada en las implementaciones de API y los recursos de administración reales.

La capa de representación

Claramente, la presentación de su API debe ser lo más sencilla posible para los desarrolladores. Al separar este elemento de la implementación, puede centrarse en crear de manera central un acceso cordial a sus API, sin impactar las API o los recursos administrativos. Esto facilita mucho más la presentación de sistemas administrativos complejos como interfaces centradas en dispositivos móviles y web que los desarrolladores pueden comprender y aprovechar rápidamente para desarrollar aplicaciones potentes y sencillas.

La capa de organización

Aunque algunas aplicaciones pueden entregar valor al acceder a un recurso único a través de una API única, las posibilidades crecen exponencialmente cuando combina datos de diversas API (incluidas algunas de otras empresas) y diversos recursos administrativos. Implementar una capa de organización junto a las interfaces puede habilitar tales combinaciones, como también simplificar el proceso de componer nuevas implementaciones de diversos recursos administrativos.

La manera más eficiente de crear una arquitectura de API centralizada es mediante la implementación de una solución de administración de API. En la séptima parte, describiremos los componentes clave de la administración de API.

Séptima parte: Administración de API

Desarrollar una infraestructura que centraliza los componentes comunes de arquitectura de API centradas en el desarrollador puede simplificar considerablemente el proceso de implementación de las API que agregan valor real a su empresa. Sin embargo, desarrollar semejante infraestructura internamente puede ser un gran reto. Afortunadamente, una gama de proveedores de software empresarial ahora ofrece soluciones de “administración de API” que quitan la necesidad de desarrollar esta infraestructura crítica dentro de la empresa.

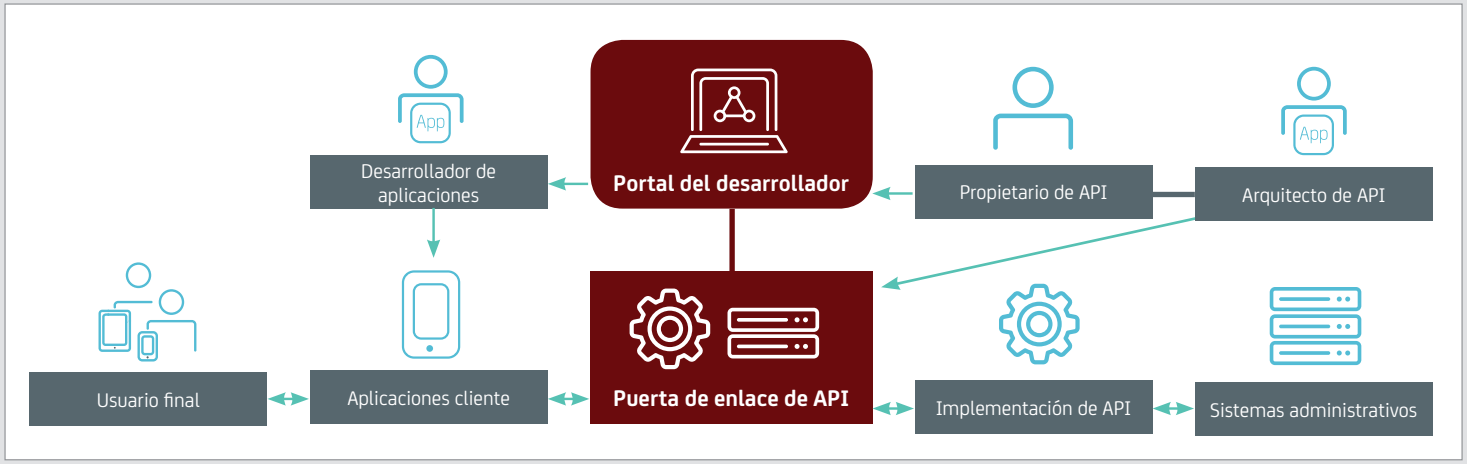
Además, como lo sugiere el nombre, las soluciones de administración de API también incluyen funcionalidades para administrar y optimizar el desempeño de las API a largo plazo. Y las soluciones más potentes también tienen funciones para desarrollar una interfaz basada en la Web a través de la cual los desarrolladores pueden descubrir, aprender sobre y acceder a las API; una parte absolutamente fundamental de la presentación de una API centrada en el desarrollador, que no se puede desarrollar en la implementación misma.

Componentes de la administración de API

Una solución de administración de API empresarial tendrá dos componentes clave:

- Puerta de enlace de API: brinda las funcionalidades de seguridad, almacenamiento en caché y organización necesarios para implementar una arquitectura de API principal.
- Portal del desarrollador: proporciona una interfaz personalizable a través de la cual los desarrolladores acceden a las API, como también a la documentación, los foros de la comunidad y otros contenidos útiles.

Figura 7: Componentes de la administración de API



Es importante observar que la administración de API no es solo un requisito técnico. Inevitablemente tendrá un papel en el éxito empresarial de cualquier programa de API empresarial. Administrar la composición, el desempeño y la seguridad de las API empresariales es fundamental para asegurarse de que la organización tenga un buen retorno de su inversión en un programa de API. De la misma manera, es fundamental atraer y administrar activamente a los desarrolladores para asegurarse de que desarrollen aplicaciones que creen valor empresarial.

Para la mayoría de las empresas, la infraestructura de la administración de API será esencial para el diseño, la implementación y el mantenimiento de las API que los desarrolladores utilizarán para crear nuevas aplicaciones realmente potentes.

Descubra lo esencial sobre la administración de API con el libro electrónico Cinco pilares de la administración de API

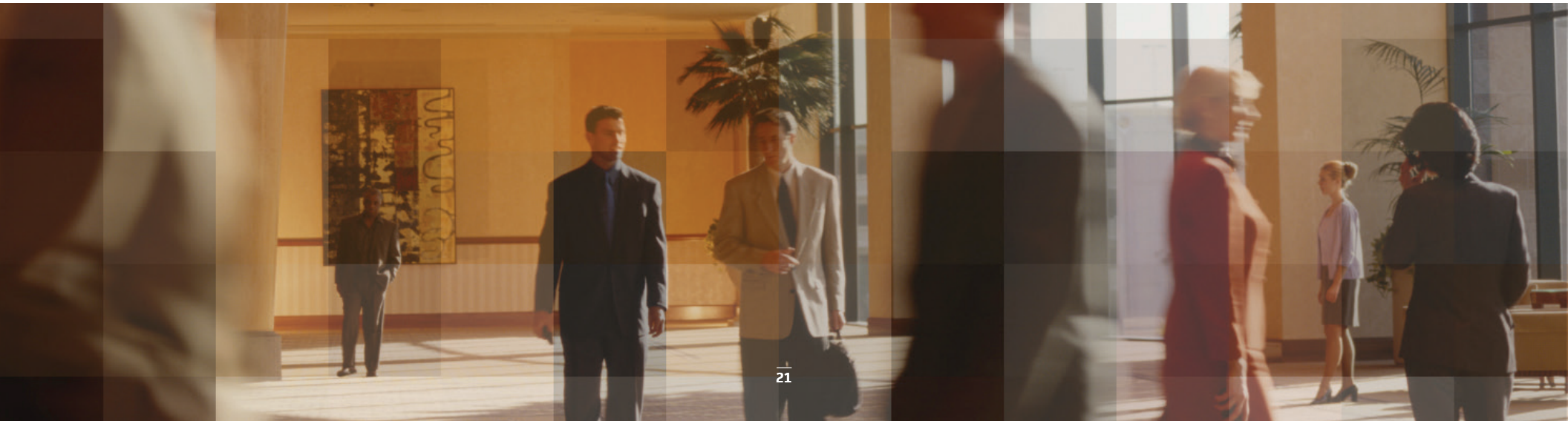
Conclusión

Desde un punto de vista arquitectónico, las API representan una extensión de SOA. De la misma manera que SOA creó interfaces para abrir sistemas heredados para volver a utilizarlos en servicios nuevos que pueden abarcar límites organizativos, las API se utilizan para abrir el sistema administrativo de la empresa a los desarrolladores que están desarrollando aplicaciones para dispositivos móviles y la Web pública. Esta es una extensión importante y es probable que los requisitos de diseño de una API web sean muy diferentes a los requisitos de un servicio web de SOA.

Mientras que los programas de SOA generalmente están impulsados por la necesidad de ahorros de costos de TI, los programas de API se centran en la generación de nuevos flujos de ingresos. Una buena API conecta una gama de activos empresariales existente para crear valor de maneras nunca antes vistas. Un buen diseño de API siempre se centra en los resultados empresariales. Por lo tanto, las prácticas de arquitectura y diseño de API deben estar alineadas con la estrategia empresarial de la organización, desde la base.

Los propietarios y arquitectos de API deben comunicarse para asegurarse de ponerse de acuerdo en los objetivos clave, cómo los alcanzarán y cómo medirán su éxito. Para garantizar que la comunicación sea efectiva, un predicador de API que pueda salvar la brecha entre los roles empresariales y técnicos debe analizar las necesidades de los líderes empresariales, propietarios de API, desarrolladores de aplicaciones y arquitectos empresariales para poder negociar un conjunto de objetivos, tareas y métricas adecuadas.

En práctica, diseñar una API para lograr éxito empresarial normalmente significa crear una interfaz que los desarrolladores realmente quieran usar. Por lo tanto, antes de desarrollar algo, es fundamental que investigue sistemáticamente su audiencia de desaroladores para poder comprender quiénes son sus desarrolladores objetivo y qué buscan en una API. También puede ser útil probar cualquier supuesto acerca de las preferencias de los desarrolladores al ofrecer API de prototipo ligero.



Conclusión

Una vez que está listo para diseñar su implementación de API real, tendrá que elegir el estilo de diseño que mejor se adapte a su proyecto. Las API de servicios web se adaptarán a programas internos dirigidos a desarrolladores con experiencia en SOA. Las API REST pragmáticas son más adecuadas para los proyectos de API abierta centrados en dispositivos móviles y la Web. Los estilos Hipermedios e Impulsado por eventos están emergiendo como estrategias que pueden ser más sostenibles en el futuro impulsado por IoT y los dispositivos móviles.

Sin importar el estilo. hay ciertos elementos arquitectónicos que todas las API deben incluir: por ejemplo, seguridad, almacenamiento en caché, representación y organización. Para lograr eficiencia y administración máximas, estos elementos no se deben desarrollar en implementaciones de API individuales. En cambio, todas las API deben impulsar una arquitectura de API en capas y central que se encuentre entre la periferia de la empresa y las mismas API.

La manera más eficiente y eficaz de implementar una arquitectura de API central (y garantizar que el programa de API permanezca siendo exitoso a largo plazo) es adoptar una solución de administración de API. Hay una variedad de soluciones en el mercado, pero la mayoría incluyen dos componentes comunes:

- Una puerta de enlace de API que proporciona la funcionalidad de seguridad y otra infraestructura clave.
- Un portal del desarrollador que simplifica el proceso de atracción y habilitación de los desarrolladores.

Hay mucho en juego en los proyectos de API empresariales de la actualidad: grandes oportunidades empresariales, riesgos de seguridad considerables y mucho más. Es fundamental que se prepare antes de comenzar a desarrollar una API: alinee los objetivos de diseño con los objetivos empresariales; establezca las preferencias de sus desarrolladores objetivo; elija un estilo de implementación adecuado; e implemente una infraestructura de administración de API. Luego, estará listo para desarrollar una API verdaderamente valiosa.

Figura 8: Prerrequisitos para lograr un buen diseño

Solo CA API Management permite que las organizaciones integren sistemas, simplifiquen el desarrollo de aplicaciones y rentabilicen datos con el nivel de seguridad y protección de API que las empresas necesitan hoy en día. Obtenga información acerca de CA API Management en **ca.com/ar/api**

Acerca de CA API Management

Con más de 300 clientes de administración de API en diversos sectores como comunicaciones, servicios financieros, gobierno y minoristas, CA Technologies ofrece tecnología líder en el sector y conocimientos que ayudan a las organizaciones a brindar valor a través de las API. CA proporciona una solución de administración de API completa, incluida la puerta de enlace de API con funciones completas y con funciones de calidad militar, además de un portal del desarrollador en las instalaciones y en versiones SaaS. Obtenga información acerca de CA API Management en **ca.com/ar/api.**

API Academy

Servicios de diseño, arquitectura y estrategia de API

El equipo de API Academy consta de expertos del sector que han sido reunidos por CA Technologies para desarrollar recursos gratuitos para la comunidad y proporcionar servicios de consultoría para organizaciones que desean llevar los programas de API al próximo nivel. Para obtener más información sobre cómo API Academy puede ayudar a su organización con la estrategia, la arquitectura y el diseño de API, visite **apiacademy.com**.

CA Technologies (NASDAQ: CA) crea un software que impulsa la transformación en las empresas y les permite aprovechar las oportunidades de la economía de las aplicaciones. El software es el centro de cada empresa, en cada sector. Desde la planificación hasta el desarrollo, la administración y la seguridad, CA trabaja con empresas en todo el mundo para cambiar el estilo de vida, realizar transacciones y comunicarse, mediante entornos móviles, de nubes públicas y privadas, centrales y distribuidos. Obtenga más información en: **ca.com/ar**.

CS200-131275

ca technologies