Gobierno de APIs. (API Owner)

Gobierno de la API

Índice del capítulo

De un vistazo

- Introducción.
- ¿Qué es el Gobierno de APIs?
- Equipo de Gobierno.
- ¿Qué hace un equipo de gobierno de APIs?

Introducción

- Las **APIs** son una **imagen externa e interna** de las **empresas** que exponen en forma de producto ciertos activos de datos o funciones definidos expresamente para su consumo a través de una **interfaz documentada y sencilla de utilizar**.

- Enlazando esta idea con el concepto de **API Economy**, mediante el cual las **empresas** impulsan su crecimiento con el **consumo** y la **monetización** de las APIs, estos productos digitales se convierten en un **activo estratégico de negocio**.

Introducción

- Resulta básico establecer una **estrategia** que favorezca la **adopción de estos componentes** de negocio, para lo cual es necesario:
  - **Involucrar a negocio**. Las APIs **no son meramente tecnología**, sino que constituyen un producto en sí mismo. Es negocio quien tiene el mayor **conocimiento funcional** y sin su respaldo las APIs son una implementación tecnológica más **sin impacto**, desaprovechando su potencial.
  - Establecer **objetivos** marcando **plazos** y usando **métricas** para controlar su cumplimiento.
  - Definir **roles** y **responsabilidades** estableciendo quién hace qué y en qué momento en el flujo de trabajo.
  - Dar **visibilidad** a la iniciativa evangelizando la **disciplina y difundiendo los estándares y buena prácticas** establecidos dentro de la organización.

¿Qué es el Gobierno de APIs?

- No es fácil encontrar una definición exacta de qué es el **Gobierno de APIs**. Podríamos decir que es una **parte muy importante** (posiblemente la más importante) dentro del proceso global de **Gestión de APIs**.

- El **Gobierno de APIs** nos va a ayudar a definir temas como **documentación**, política de **versionado**, **seguridad**, **diseño**, **monitorización**, **testing**, etc., es decir, va a estar presente a lo largo de todo el **ciclo de vida** de nuestras APIs.

- Los **objetivos principales** del Gobierno de APIs son la **estabilidad** y la **coherencia**.

- Con la **estandarización** de muchos **procedimientos** dentro del **ciclo de vida** de una API hará que nuestra organización cuente con una cierta coherencia que eliminará **muchos problemas a corto/medio plazo**.

- Si mantenemos dicha **coherencia en nuestras APIs** y usamos determinadas herramientas y metodologías, además, dotaremos a nuestro negocio de **una estabilidad vital**.

¿Qué es el Gobierno de APIs?

- Definición:
  - Es el proceso **más importante** dentro de la **gestión de APIs**.
  - Su **objetivo principal** es **estandarizar** los procedimientos dentro del **ciclo de vida** para garantizar estabilidad y coherencia en las APIs.
  - El **gobierno de APIs** se encarga de definir los **estándares a utilizar** dentro de la empresa para temas como documentación, herramientas tecnológicas, políticas de versionado y escalado; monetización, pruebas, publicación en el catálogo de APIs, seguridad, monitorización, nombramiento de endpoints, códigos de respuesta y manejo de errores.

Equipo de Gobierno

- Un **Equipo de Gobierno** es un conjunto de personas de **diferentes perfiles** encargadas de velar por la **estabilidad y coherencia del API Management** de una organización.

- En muchos casos, esta parte **no está muy bien definida** y no queda claro **quién/es son los encargados** de formar este equipo porque en algunas ocasiones ya forman parte de otros equipos.

- Es recomendable que la gente que forme parte de dicho equipo tenga **diferentes perfiles**: arquitectura, experto funcional (Business Analyst), seguridad, technical leader, desarrollo, etc. Es decir, gente con **ciertas habilidades** de gestión, comunicación, liderazgo y gente con **habilidades más técnicas** (API Specialist, API Evangelist).

Equipo de Gobierno

- El **objetivo** de un Equipo de Gobierno será:
  - Gestionar todo el **ciclo de vida** de una API.
  - Establecer unas **buenas prácticas** dentro del diseño e implementación de APIs (evangelización).
  - Establecer una **metodología de trabajo óptima** a la hora de la Gestión de APIs.

¿Qué hace un equipo de gobierno de APIs?

- Fomentar la **transparencia**:
  - Se encarga de recopilar el conocimiento en el **catálogo de APIs** y se convierte en el único punto de entrada para diseñar, organizar y publicar las APIs en dicho catálogo.
  - **Comunica y forma** a la organización, independientemente del perfil (negocio o tecnología), en todo el proceso colaborativo del modelo de gobierno.
- Estandarización:
  - El core del trabajo desarrollado por el equipo de gobierno es establecer un **conjunto de buenas prácticas** mediante guías de desarrollo e implementación que tienen como objetivo **homogeneizar** el diseño y la **documentación de las APIs** y facilitar la **toma de decisiones** en todos los procesos asociados a ellas estableciendo un marco de trabajo común.

¿Qué hace un equipo de gobierno de APIs?

- **Definir roles y responsabilidades**:
  - El gobierno es el encargado de identificar a los **diferentes participantes** e involucrarlos en el flujo de trabajo, estableciendo **en qué momento** del flujo interviene cada participante y las **tareas** a realizar en ese punto.
- **Gestionar métricas**:
  - Define los **controles** y mecanismos de medida mediante KPIs de ámbito técnico para comprobar el **uso** y **buen funcionamiento** de las APIs y de ámbito de negocio para cuantificar el éxito y acogida por parte de los desarrolladores que consumen la API.

¿Qué hace un equipo de gobierno de APIs?

- **Definir la metodología de trabajo**:
  - El equipo de gobierno se ocupa de establecer el **flujo de trabajo** y **mejorarlo** de manera continua gracias a las métricas. También define los **canales de comunicación** y las ceremonias necesarias (reuniones, comités, dailys, etc) que permiten **articular el flujo de trabajo**. Además, incluye puntos de **control** para garantizar el cumplimiento de las buenas prácticas.
- **Evangelizar** a la organización:
  - Impulsa la adopción de **soluciones** que incorporen APIs y promueve la reutilización de las ya existentes.
  - Garantiza el **alineamiento estratégico con negocio** y ayuda en la identificación de activos "APIificables" que aporten valor a la organización.
  - También se ocupa de evangelizar sobre el **trabajo en comunidad y dinamizarla**.

Gobierno del ciclo de vida de un API

- Introducción.
- Inicio.
- Fase de diseño.
- Fase de desarrollo.
- Uso o consumo del API.
- Monitorización.
- Versionado y eliminación.

Introducción

- Ahora vamos a ver desde el inicio hasta el final del ciclo de vida de una API y cómo interviene (o debería) un Equipo de Gobierno y los beneficios que tiene.
- Sus fases son:
  - Inicio.
  - Fase de diseño.
  - Fase de desarrollo.
  - Uso o consumo del API.
  - Monitorización.
  - Versionado y eliminación.

Inicio

- Tras los aspectos más comerciales se inicia con **una(s) reunión(es) de Kick off** donde están presentes varios perfiles y donde tiene que estar obligatoriamente el Equipo de Gobierno.
- Es posible que **no tengan que asistir todos** los miembros (perfiles) a esta reunión.
- Puede que con el **Responsable, Arquitecto y experto funcional** sea suficiente por ahora en estas primeras reuniones.
- Se analizarán de las **necesidades de negocio** así como las posibles interacciones entre **APIs internas/externas** ya **existentes o nuevas** a desarrollar.

Inicio

- Puede que lo que se **necesite** ya esté implementado en la actualidad pero el **desconocimiento** de que APIs están disponible puede **hacer trabajar por duplicado**.
- Por ello se tienen que tener **catalogados todas las APIs de una organización** (API Discovery).
- En este **catálogo** tienen que aparece **todas las APIs** junto con su **documentación** tanto funcional como técnica.
- Ya veremos más adelante cómo se puede implementar dicho catálogo de forma más eficiente pero la gran mayoría de los API Managers tienen un API Portal o Portal del desarrollador.

Inicio

- **Posibles problemas**:
  - El **equipo de seguridad** no está involucrado en estas **reuniones** y se detectan **vulnerabilidades** después de la integración entre APIs.
  - **No existe un catálogo** actualizado de APIs internas y se duplican APIs (servicios).
  - **Rendimiento** esperado del API (servicio) no está claro o indeterminado (necesidad de escalado).

Fase de diseño

- Es fundamental aplicar una serie de **buenas prácticas** en cuanto al diseño de nuestras APIs con el fin de estandarizarlas y establecer una coherencia entre ellas.
- Desde como **interactúan** diferentes APIs entre sí, el **naming**, correcto uso de los **verbos** y códigos de respuesta, gestión de errores, etc.
- Una vez analizado todo esto en profundidad con los perfiles más técnicos se puede llegar a un **acuerdo de interfaz (Contract First)**.
- Esto ayuda a garantizar que **las APIs sean consistentes** y **reutilizables**.

Fase de diseño

- Si esto se cumple **no vamos a encontrarnos con APIs** cuyos verbos no son los correctos, los códigos de respuesta **no son los adecuados** o el **naming** no es el mismo para todos.
- ¿A qué os ha pasado esto alguna vez? ¿El Equipo de Gobierno ha estado presente en el diseño del API?
- Si hubiese estado todas las nuevas APIs tendrían una **definición y diseño homogéneo**.
- El cómo se **implemente** ya es otra cuestión pero, por lo menos, se tiene **un catálogo actualizado** con un **diseño común a todas las APIs** de la organización por lo que la **integración entre APIs** siempre va a ser mucho más sencilla **reduciendo tiempos y problemas**.

Fase de diseño

- En este punto se recomienda invitar a miembros de otros equipos a participar activamente con el fin de evangelizar, es decir, poner encima de la mesa todas estas buenas prácticas para que se vayan distribuyendo por todos los equipos.

- No se tiene que quedar en **“el Equipo de Gobierno me dice cómo tengo que llamar a mi nueva API, los verbos, códigos de respuesta y me define el acuerdo de interfaz"**.

- El **Equipo de Gobierno** tiene como objetivo **hacer entender al resto de equipos** el porqué se toman las decisiones que se toman en cuanto al diseño se refiere.

Fase de diseño

- Posibles problemas:
  - Acuerdo de interfaz **poco sólido o indeterminado** (problemas en la integración final). Cuanto **más cariño** se dé a esta parte entre todas las partes **menos problemas** vamos a encontrarnos en el proceso de integración.
  - Herencia de **malas prácticas** con verbos, naming, códigos de respuesta, etc.

Fase de desarrollo

- **Documentación**:
  - Una vez que se tiene claro la interfaz, se pasa a la **parte de la documentación de las APIs** donde el Equipo de Gobierno tiene que determinar una **serie de pautas y herramientas** para llevar a cabo este paso.
  - Dentro de **todas las posibilidades** que se tienen en el mercado para documentar APIs tienen que tomar la decisión de **elegir la qué mejor se adapte a sus necesidades**:
    - **Swagger**,
    - **RAML**,
    - Stripe, API Blueprint, etc.
  - Lo habitual es que **está decisión no dependa de una API** en **concreto** sino que sea una decisión previamente tomada **a nivel corporativo**. Posiblemente se necesitará crear usuarios, permisos, roles para empezar a hacer la documentación de nuestro API.

Fase de desarrollo

- **Implementación**:
  - Muchos pueden pensar que la fase de implementación **no forma parte del Gobierno de APIs** pero siempre es bueno, tal y como se ha hecho con el diseño, que en la medida de lo posible homogenizar la **implementación** a nivel de **frameworks, librerías, clientes, timeouts**, etc.
  - En esta fase evidentemente toman **protagonismo** los **perfiles más técnicos** del Equipo de Gobierno que se deberían encargar de formar, si es necesario, y establecer una **serie de buenas prácticas para la implementación del API**.

Fase de desarrollo

- **Implementación**:
  - Posibles **problemas**:
    - Documentación **incompleta o no actualizada**. Esto va a provocar una implementación incorrecta, por lo tanto, problemas en la **integración**.
    - ¿Quién es el **propietario** de la documentación? ¿El Equipo de Gobierno? ¿El equipo de desarrollo?
      - El Equipo de Gobierno toma parte en esta fase pero lo normal es que el propietario de la documentación del API sea **el equipo de desarrollo**.
      - Esta parte tiene que quedar **muy clara** ya que el **propietario** será el encargado del **mantenimiento** de dicha documentación (actualizaciones, versionado, etc.).

Uso o consumo del API

- Esta fase de **uso o consumo del API** es vital, posiblemente sea la fase donde **más riesgos** podemos encontrar si no se hace correctamente el **Gobierno de APIs**.
- Una API está hecha para ser **consumida**, bien sea **interna o externamente**, para ello aparece el rol del **consumidor** que puede ser otra API, un frontal (web, apps, etc) o cualquier otro componente.
- Es vital saber en todo momento **qué consumidores** tienen nuestras API.
- Los consumidores se podrán **subscribir** a nuestras APIs desde el **API Portal o Portal del desarrollador**.
- Esta **subscripción** podrá ser **libre o bajo demanda**, según la tipología tanto de la API como el consumidor, es decir, puede que para **nuestra organización** no dejemos que **cualquier consumidor** se subscriba libremente a nuestras APIs sino que alguien del Equipo de Gobierno o desarrollo confirme dicha subscripción.

Uso o consumo del API

- Hay que tener presente en todo momento la **capacidad de nuestras APIs** (servicios), es decir, **cuántas peticiones por segundo** son capaces de procesar (máximas y esperadas).

- Y aquí es donde aparece otro concepto importante como **las cuotas**, es decir, **qué cantidad de peticiones por segundo** voy a otorgar como máximo a un consumidor para que haga uso de una determinada API.

- Para ello, como hemos dicho anteriormente es vital analizar previamente **el rendimiento de nuestra API** y en función de eso, de la tipología del API, del tipo de consumidor (interno o externo) determinar qué **nivel de servicio** o **cuota** le vamos a dar para el consumo de nuestra API.

- Cada **consumidor** tiene que tener **varios responsables** (funcional y técnico) y una forma de contacto (preferiblemente un buzón de correo electrónico). Todo **cambio de una API** tiene que ser comunicado a **todos sus consumidores** (nuevas versiones, cambios retrocompatibles, versiones deprecadas, versiones eliminadas, cambios en la documentación, etc.).

Uso o consumo del API

- Posibles **problemas**:
  - Se **desconoce** realmente **quién consume** nuestras APIs y qué versión.
  - Si se desconoce este hecho tenemos un **problema bastante importante,** ya que no vamos a poder deprecar o eliminar versiones **sin riesgo**, y es justo una palabra que se tiene que evitar usar cuando se trata de Gobierno de APIs.
  - Por ejemplo, imaginaos que nuestro negocio trata de la logística de órganos para transplantes entre diferentes hospitales. Si no sabemos al 100% qué consumidores tiene mi API ¿se podría eliminar una versión/API dejando a un **hospital sin este servicio**?

Uso o consumo del API

- Posibles **problemas**:
  - Mala gestión de cuotas. No se analiza en detalle ni la **tipología** del API ni del **consumidor** y se establecer cuotas por defecto.
  - Esto puede provocar la **saturación o incluso caída del servicio**.
  - Por ejemplo: no analizamos a fondo el rendimiento de nuestra API ni tampoco prestamos atención a los consumidores estableciendo cuotas máximas para todos. Damos la **misma cuota** a un consumidor de una **empresa pequeña** que una **empresa internacional** que está pagando (monetización) por el servicio.
  - Puede que la empresa pequeña **pagando 100 veces menos** que la empresa **grande** haga un **uso elevado de nuestra API** y la empresa grande vea **ralentizadas** sus peticiones o, incluso, se provoque una **caída del servicio con la pérdida de dinero** y **prestigio** que conlleva.

Uso o consumo del API

- Posibles **problemas**:
  - Se usa un **único email de responsable técnico** (no un buzón) para un consumidor y tras el paso de los meses esa persona sale de la empresa por lo que producen errores en las notificaciones de las APIs consumidas.

Monitorización

- Mediante la **monitorización** vamos a saber **el consumo real** de nuestras APIs.
- No sólo si los consumidores están consumiendo realmente sino la forma en la que lo hacen (picos, franjas horarias, etc).
- **Monitorizando** nuestras **APIs** vamos a poder tener un elemento importante para poder **refinar las cuotas** a aplicar a los consumidores.
- También nos va a servir para **determinar o modificar políticas de escalado** de nuestros servicios.

Monitorización

- Gracias a la monitorización vamos a poder generar **estadísticas e informes** a diferentes departamentos de nuestra organización como IT, Negocio, Marketing, etc.

- Además, nos va a servir para implementar **algún sistema de alertas** que notifique algún comportamiento **anormal o inusual** del consumo de nuestras APIs.

Versionado y eliminación

- Otra de las partes **fundamentales** a tener en cuenta en el ciclo de vida de un API es la que tiene que ver con el **versionado y la eliminación** de determinadas versiones/APIs.
- Es necesario **notificar** a todos los **consumidores** de cualquier cambio relacionado con el API y/o documentación.
- Es necesario enviar un **fichero changelog** con los cambios así como la documentación si fuese necesario.

Versionado y eliminación

- El **Equipo de Gobierno** en la fase Inicial y/o Diseño tiene que determinar **cuántas versiones** se mantienen activas.
- Para ello es importante saber el **TTL de una versión**, es decir, el **tiempo** que una versión permanece activa en producción.
- Puede que por la **tipología de la API** se publiquen nuevas versiones cada **2 meses, cada 6 meses o cada 2 años** por ejemplo.
- Esto afecta directamente al **número de versiones** que se esperan **tener activas** pudiendo ser un problema tanto para quien hace la implementación como para quién consume la API.

Versionado y eliminación

- Este análisis inicial puede **cambiar en el futuro** pero nos puede servir para informar a los consumidores de nuestras APIs (generalmente externos) sobre este proceso de versionado y eliminación con el fin de tener en cuenta posibles procesos de migración.

- Esta parte es más **comprometida** ya que, en muchas ocasiones, podemos encontrarnos con **integraciones de terceros** «a proyecto cerrado«, es decir, se integran con una **determinada versión** y luego no tienen la capacidad (de equipo, de presupuesto, etc.) de migrarse a **versiones superiores**.

Versionado y eliminación

- **Posibles problemas**:
  - Una **mala definición** del API y/o servicio puede provocar la necesidad de crear **demasiadas versiones** en muy poco tiempo.
  - Tener muchas versiones activas conlleva una **sobregestión** y problemas de **mantenibilidad** en código.
  - Puede ocurrir que **determinados consumidores** no puedan integrarse con las **nuevas versiones** (temas contractuales, disponibilidad, etc.). ¿Y ahora qué hacemos? Si una empresa (consumidor) se ha integrado con la v1 de nuestra API y cuando deprecamos esta versión luego no podemos eliminarla en el futuro porque la empresa no tiene necesidad ni capacidad para migrarse de versión. El **Equipo de Gobierno** tiene que tener esto siempre en mente y establecer de alguna forma una **política de versionado/eliminación** consensuada con los **consumidores**.
  - Se versiona el **código** (API y lógica) pero **no se versiona la documentación**.

Estandarización

- El **objetivo de la estandarización** es garantizar que la API sea fácil de **comprender** y **utilizar** para los desarrolladores que la consumen.
- Al seguir **estándares comunes y reconocidos**, se asegura que la API sea **intuitiva y accesible** para una amplia comunidad de desarrolladores, lo que puede mejorar la adopción y el uso de la API.
- Además, la **estandarización** también puede ayudar a mejorar la **calidad de la API** y a simplificar el proceso de desarrollo. Al seguir un conjunto de **reglas claras y estandarizadas**, se asegura que la API cumpla con los estándares de calidad y seguridad requeridos, y se facilita el proceso de pruebas.

Testing

- Introducción.
- ¿Qué son las pruebas de API?
- Tipos de Pruebas de API.
- Herramientas para las pruebas API.
- Beneficios de las pruebas API.

Introducción

- Las **pruebas de API** suponen realizar operaciones, tales como, **enviar** peticiones a una API, **observarlas** y **monitorizar las respuestas** para asegurarse de que funcionan de la forma que se espera.

- Las **pruebas de API** son una parte fundamental del **ciclo de vida del desarrollo de las APIs** y tienden a evaluar la **usabilidad, la fiabilidad, la eficiencia y la seguridad** de una API.

- Las **pruebas de API** se concentran en la **funcionalidad** de una aplicación más que en cada una de sus partes o **cómo se muestran** a un usuario. Estas últimas, en general, se llaman **pruebas de IU** (interfaz de usuario). A continuación, se incluye una **breve descripción de las diferencias** entre pruebas de API y IU.

Introducción

- **Pruebas de API vs. Pruebas de IU:**
  - En las pruebas de API, los usuarios utilizan **software** que realizan **peticiones a la API**, **recuperan** los resultados y **guardan la respuesta** del sistema en vez de las entradas y salidas de los usuarios.
  - Las **pruebas de API** se centran en la **lógica de negocios** del software y no tanto en **cómo se ve** y funciona una aplicación. Sin embargo, si se utiliza una **interfaz gráfica de usuario** (GUI por su sigla en inglés), las **pruebas en la interfaz de usuario** encuentran defectos en los productos y el software.
  - Al probar pantallas y elementos, tales como, **menús, botones e iconos**, las pruebas de IU tienen como objetivo asegurarse de que el funcionamiento de las funciones de la aplicación de software sea correcto de acuerdo a los estándares establecidos.

Tipos de Pruebas de API

- Pruebas de **Integración**:
  - Las **pruebas de integración**, pruebas de String (o Pruebas de Hilos) comprueban si los **componentes de un software** están conectados de **forma lógica** y los examinan como una unidad. Los distintos componentes de software que son creados por varios programadores generalmente forman una unidad de software. El objetivo de este tipo de pruebas es encontrar problemas con **cómo se comunican** los distintos componentes del software **cuando trabajan juntos**.
- Pruebas de **Rendimiento**:
  - Las **pruebas de rendimiento** son un enfoque para evaluar software que mide la **capacidad** de **respuesta, escalabilidad, estabilidad** y utilización de recursos de una aplicación en respuesta a una carga de trabajo determinada.
  - Encuentran y arreglan problemas de **rendimiento** de aplicaciones de software. Encuentran qué se debe cambiar en un software antes de que se lance a producción. Sin las pruebas de rendimiento, el producto podría tener problemas tales como rendimiento muy lento si varios usuarios lo utilizan al mismo tiempo.

Tipos de Pruebas de API

- **Pruebas de Carga**:
  - Las pruebas de carga evalúan la funcionalidad de un programa de software cuando **se somete a distintas cargas**. Es un procedimiento de pruebas no funcionales que se lleva a cabo utilizando un tráfico predeterminado y esperado. Se realizan antes de que se lance una aplicación, y tienen como objetivo **eliminar los cuellos de botella** que puedan existir y garantizar la **estabilidad y la operación eficaz** de las aplicaciones de software.
- **Pruebas de Seguridad Dinámica**:
  - Las pruebas de seguridad buscan encontrar los **problemas que puede tener un sistema** y definen si los recursos y los datos de un programa están protegidos de los hackers. Se aseguran de que el software está a salvo de cualquier peligro o amenaza que podría resultar dañina. Las pruebas de seguridad sobre un sistema tienen como objetivo identificar los posibles problemas y las debilidades que podrían llevar a que se perdieran datos y se dañara la reputación de una empresa.
  - Las **pruebas dinámicas de seguridad para las API** replican un ataque real e identifican los riesgos que podrían haberse generado tanto en el código que fue escrito por el equipo de desarrollo como en las dependencias con librerías de código abierto.

Tipos de Pruebas de API

- **Pruebas Fuzz**:
  - Las pruebas “fuzz” de API son una **técnica de pruebas automáticas** que implican enviar **información incorrecta** a una API para comprobar si las aplicaciones fallan o hay algún otro problema. El fuzzing encuentra problemas y errores ocultos.
  - Fuzzing es una técnica que generalmente utilizan los hackers para encontrar errores en programas por más que tengan el código. Los hackers pueden potenciar las vulnerabilidades que hayan encontrado para atacar las API con DDoS e inyectar SQL, entre otras cosas.
  - Las pruebas fuzz, sin embargo, no ofrecen una visión completa sobre el estado de seguridad de una API. Se necesitan más pruebas API para tener al mayor nivel posible de seguridad para las APIs.

Tipos de Pruebas de API

- **Pruebas de Validación**:
  - Las pruebas de validación confirman si el software que ha sido sujeto a pruebas y que ha sido desarrollado cumple con los **requisitos del usuario**.
  - Se deben llevar a cabo **pruebas orientadas a detalles** sobre los **escenarios** y la lógica de las **especificaciones** de requisitos.
  - Las **funciones fundamentales** de una aplicación deben ser examinadas en esta área.
  - El tester debe definir si los resultados de la ejecución de las pruebas son **consistentes** con la información en la especificación de requerimientos.

Tipos de Pruebas de API

- **Pruebas de Interoperabilidad**:
  - Las pruebas de software de este tipo determinan si un programa **se puede comunicar con otros** programas y sistemas informáticos.
  - Las pruebas de interoperabilidad se aseguran de que **no haya problemas de compatibilidad** en la capacidad del producto de software de **interactuar con otras partes** del software.
  - En otras palabras, las pruebas de interoperabilidad implican probar que el **funcionamiento end-to-end** de la comunicación bidireccional entre los sistemas cumple con los requerimientos.
  - Por ejemplo, las pruebas de interoperabilidad entre tabletas y teléfonos inteligentes verifican que se puedan transferir datos vía Bluetooth.

Tipos de Pruebas de API

- Pruebas de Runtime y Errores:
  - Este tipo de pruebas analizan **cómo operan las APIs**. Particularmente, el resultado general de aplicar el código de una API.
  - Este método se enfoca en una de las siguientes áreas: detección de errores, seguimiento, ejecución de errores, o desviaciones de recursos.
  - La detección **errores en tiempos de ejecución** puede ayudar a identificar las **principales razones** por las cuales una aplicación falla, funciona mal, o se comporta de forma inesperada, ya que identifica errores que sólo ocurren en tiempo de ejecución.

Tipos de Pruebas de API

- Pruebas de Timeout:
  - Si a un servicio le lleva **más tiempo del esperado** responder a una solicitud, ocurren **errores de timeout** (tiempos de espera de conexión).
  - El **timeout** es una configuración que puede **cambiarse** dependiendo de la plataforma. Si ocurre un timeout, la respuesta contendrá información además del **código de estado 500**. Puede haber varias causas para los timeout, entre ellos **demasiados datos en la solicitud** o un problema con la red o el servicio.
  - Puede ser necesario probar el comportamiento del mecanismo de seguridad de la aplicación; es decir, cómo actúa cuando sistemas externos no responden dentro del tiempo que se espera que lo hagan. Por ende, es recomendable que se hagan pruebas relacionadas al timeout.

Herramientas para las pruebas API

- Las pruebas sobre APIs son **fundamentales** para realizar **pruebas efectivas**.
- El **equipo de desarrollo** va a necesitar **automatizarlas** si quieren ser más productivos.
- Si se utilizan herramientas para pruebas APIs se pueden evitar los **problemas** y **errores recurrentes** que involucran las **pruebas manuales**.
- Además, generalmente se puede combinar una **herramienta de automatización de pruebas APIs** con el **flujo de trabajo de integración continua**.
- Esta integración es una **muy buena opción** para mejorar la calidad de tu código y para encontrar **defectos durante el ciclo de vida** de desarrollo del software.

Herramientas para las pruebas API

- **Postman**:
  - Puedes utilizar Postman para supervisar la API, para crear pruebas automatizadas, para llevar a cabo depuraciones, y para enviar peticiones. Actualmente, Postman ha ampliado las funciones que tiene para Windows y macOS. Postman ha incluido funciones muy innovadoras en su última versión para mejorar las pruebas, la colaboración, las integraciones, y también la gestión y seguridad de la API. Es una herramienta sin costo, pero hay planes pagos que incluyen funciones avanzadas.
- **Playwright**:
  - Playwright puede obtener acceso a la REST API de la aplicación. En algunas ocasiones, puedes querer utilizar Node.js para enviar peticiones directamente al servidor en vez de cargar una página web y ejecutar en ella código JavaScript. Esto puede suceder para probar tu API, para configurar el lado del servidor antes de acceder a la aplicación, o para verificar las condiciones del servidor antes de ejecutar ciertas actividades relacionadas con un navegador. Los métodos de APIRequestContext de Playwright son de gran utilidad para lograrlo.

Herramientas para las pruebas API

- **SoapUI:**
  - SoapUI permite que se ejecuten pruebas en las API de SOAP y en los servicios web RESTful. Es una de las herramientas más elegidas por los testers ya que permite que los usuarios diseñen escenarios complejos y facilita las pruebas asíncronas. También se destaca por su excelente ambiente para pruebas determinadas por datos y por la facilidad de su uso. Sólo uno de los dos planes de SoapUI es gratis. La versión gratis es relativamente más fácil de usar, y los testers pueden reusar sus scripts. La opción de pago soporta integraciones IC/ID, extracciones de datos de origen, y creación de código personalizado.
- **Assertible**:
  - Los desarrolladores y los testers generalmente confirman que Assertible es una de las mejores soluciones de pruebas API ya que es muy confiable. Permite pruebas API en cualquier etapa del software, desde la integración continua hasta el pipeline de entrega. Assertible ofrece muchas funciones muy útiles a sus usuarios. Se puede integrar con GitHub. Assertible también utiliza librerías de aserciones para validar respuestas HTTP.
  - Assertible tiene un plan sin costo para probarlo y aprender a utilizarlo. Además, ofrece varios otros planes de pago.

Herramientas para las pruebas API

- **Apigee**:
  - Apigee es una herramienta de pruebas API hecha en JavaScript muy interesante que le da la opción a los desarrolladores y testers de acceder a sus funciones para utilizar una gran variedad de funciones de edición. La herramienta es apropiada para APIs que contienen muchos datos, por lo que es la mejor opción para empresas digitales grandes y complejas. Al examinar el volumen de la API, el índice de respuestas, y los potenciales índices de error, encuentra problemas que pueden llegar a tener un impacto sobre la aplicación.
  - Apigee tiene una prueba gratuita que permite probar el programa. Pasado ese tiempo, las compañías y los testers deben elegir el plan que más se adapte a sus necesidades para seguir utilizándolo. Los planes que están disponibles actualmente son la prueba gratuita, el Estándar, el de Empresas, y el plan Plus para Empresas.

Herramientas para las pruebas API

- API Fortress:
  - API Fortress permite desarrollar y automatizar las pruebas de rendimiento y las pruebas funcionales de forma más fácil. Las funcionalidades incluyen una simulación continua de API utilizando plug-and-play, soporte para servicios web, y la colaboración en equipo. Es por ello que es la herramienta más efectiva para verificar SOAP y REST.

Beneficios de las pruebas API

- Reduce los costes de las pruebas:
  - Ya que las pruebas API se pueden implementar de forma más rápida, la utilización de recursos es más efectiva y los costos relacionados con las pruebas son más baratos. Las pruebas de API se pueden ejecutar antes de que se haga cualquier prueba en la IU e incluso antes de que se haya establecido la lógica del negocio. Por ende, puede ayudar a identificar problemas de forma temprana. Si se identifican problemas de forma temprana, se reducen los costos para actualizar aplicaciones y la resolución de problemas se vuelve menos cara.
- Resolución de problemas más rápido:
  - Será más fácil ver de inmediato si los resultados de las pruebas de API son positivos o negativos porque las pruebas API trabajan con rapidez para entregar una solución. Pueden ayudar a resolver problemas en etapas más tempranas del desarrollo de un sistema, y permiten que se encuentren errores y se eliminen de forma más rápida.

Beneficios de las pruebas API

- Liberaciones más rápidas:
  - Es un hecho que las pruebas de IU retrasan la entrega de productos. La opción más rápida, sin embargo, que puede ahorrarte casi ocho horas, son las pruebas de API. Ahora puedes concentrarte en otros elementos fundamentales para el desarrollo de tu software.
- Entienden cualquier lenguaje de programación.
  - Cualquier lenguaje de programación sirve para realizar pruebas de API ya que se envían datos utilizando los estándares de XML y de JSON. Por ende, no importa si prefieres utilizar Java, JavaScript o Python.
- Integración con pruebas de IU:
  - Se pueden lograr pruebas de integración con las pruebas de API. Son muy útiles si se quiere probar la IU antes de probar la API. Por ejemplo, una integración simple podría implicar agregar nuevos usuarios a la aplicación antes de que comiencen las pruebas de IU.

Seguridad

- Introducción.
- ¿Qué es la seguridad de la API?
- SOAP vs REST.
- Prácticas recomendadas.
- Gestión y seguridad de las APIs.

Introducción

- Las **empresas** utilizan las **APIs para conectar los servicios y transferir datos**.

- Las APIs **dañadas, expuestas o pirateadas** son la causa de las principales vulneraciones de la seguridad de los datos. Exponen datos **médicos**, **financieros** y **personales** que entonces quedan disponibles al público. Dicho esto, no todos los datos son iguales ni tampoco deben protegerse de la misma manera. El método que utilice para abordar la seguridad de las API dependerá del tipo de datos que deba transferir.

- Si su **API** se conecta a **una aplicación de terceros**, necesitará saber cómo dicha aplicación transfiere la información a Internet. Si tomamos el ejemplo anterior, tal vez no le importe que alguien sepa qué hay en su refrigerador, pero si la persona utiliza la misma API para rastrear su ubicación, eso sí podría ser motivo de preocupación.

¿Qué es la seguridad de la API?

- La **seguridad de las API** es el **proceso de protegerlas** de los ataques. Al igual que las aplicaciones, las redes y los servidores pueden ser objeto de ataques, las **APIs pueden ser víctimas de diferentes amenazas**.

- La seguridad de las API es un **componente central** de la seguridad de aplicaciones web.

- La mayoría de las aplicaciones web modernas **se basan en APIs** para funcionar, y las APIs introducen un **riesgo adicional** en una aplicación al permitir que partes externas accedan a ella.

- Se puede comparar con un **negocio que abre su oficina al público**: tener más personas en las instalaciones, algunas de las cuales pueden ser **desconocidas** para los empleados del negocio, introduce un **mayor riesgo**. Del mismo modo, una **API permite que personas ajenas** a la empresa utilicen un programa, lo que introduce un **mayor riesgo en la infraestructura** del servicio API.

SOAP vs REST

- La **seguridad de las API** web se ocupa de transmitir los datos a través de las APIs que están conectadas a Internet.
- **OAuth** (Open Authorization) es el **estándar abierto** para la delegación del acceso. Permite que los usuarios otorguen acceso a los recursos web **a terceros**, sin necesidad de **compartir contraseñas**.
- La mayoría de las implementaciones de API son de REST (transferencia de estado representacional) o SOAP (protocolo simple de acceso a objetos).

SOAP vs REST

- Las **APIs de REST** utilizan **HTTP** y admiten el **cifrado de seguridad de la capa de transporte** (TLS).

- **TLS** es un estándar que mantiene **privada** la **conexión a Internet** y verifica que los datos enviados entre dos sistemas (entre dos servidores, o un servidor y un cliente) estén **cifrados** y no se modifiquen. Eso significa que, si un pirata informático intenta obtener la información de su tarjeta de crédito desde un sitio web de compras, no podrá leer ni modificar sus datos. Para saber si un sitio web está protegido con TLS, solo necesita verificar que la URL comience con "**HTTPS**" (Protocolo seguro de transferencia de hipertexto).

- Las **API de REST** también utilizan notación de **objetos JavaScript (JSON)**, que es un formato de archivo que **facilita** la **transferencia de datos** a través de los exploradores web. Al utilizar HTTP y JSON, las API de REST no necesitan almacenar o volver a agrupar los datos, lo cual las vuelve mucho **más rápidas que las API de SOAP**.

SOAP vs REST

- Las **APIs de SOAP** utilizan protocolos integrados conocidos como **la seguridad de los servicios web** (WS Security).
- Esos **protocolos** definen un conjunto de reglas que se basa en la **confidencialidad** y la **autenticación**.
- Las API de SOAP son compatibles con los estándares establecidos por los dos principales organismos internacionales: la Organización para el avance de los estándares de información estructurada (OASIS) y el Consorcio World Wide Web (W3C).
- Utilizan una combinación de **cifrado XML**, **firmas XML** y **tokens SAML** para verificar la autenticación y la autorización.
- En general, las APIs de SOAP reciben un gran reconocimiento por sus **medidas de seguridad** más integrales, pero también necesitan una **mayor gestión**. Por eso se recomiendan las APIs de SOAP para las empresas que manejan **datos confidenciales**.

Prácticas recomendadas

- Estas son algunas de las formas más comunes en las que puede fortalecer la seguridad de sus API:
  - **Utilice tokens:** configure **identidades** confiables y controle el acceso a los servicios y a los recursos utilizando los **tokens** asignados a dichas identidades.
  - **Utilice métodos de cifrado y firmas**: cifre sus datos mediante un **método como TLS** (consulte la información que se presentó anteriormente). Solicite el **uso de firmas** para asegurar que solamente los usuarios adecuados descifren y modifiquen sus datos.
  - **Identifique las vulnerabilidades**: mantenga actualizados sus elementos de sus APIs, sus **controladores**, **redes** y su **sistema operativo**. Manténgase al tanto de cómo funciona todo en conjunto, e identifique las **debilidades** que se podrían utilizar para entrar en sus API. Utilice analizadores de protocolos para detectar los problemas de seguridad y rastrear las pérdidas de datos.

Prácticas recomendadas

- (cont.):
  - **Utilice cupos y límites:** establezca un **cupo en la frecuencia** con la que se puede recurrir a su API, y dé seguimiento a su historial de uso. Encontrar **más solicitudes** a una API puede indicar un **abuso**. También podría ser un **error de programación**, como una solicitud a la API en un **bucle sin fin**. Establezca reglas de limitación para proteger sus API de **ataques de denegación de servicio** y **picos de uso**.
  - **Utilice una puerta de enlace de API**: las puertas de enlace de API funcionan como el **principal punto de control para el tráfico** de las API. Una buena puerta de enlace le permitirá **autenticar** el tráfico, así como **controlar y analizar** cómo se utilizan sus APIs.

Gestión y seguridad de las APIs

- Por último, la seguridad de las APIs depende de una buena gestión. Muchas plataformas de gestión de las API admiten **tres tipos de esquemas** de seguridad:
  - **Clave de API**: una cadena de un **solo token** (es decir, un dispositivo de hardware pequeño que brinda información de autenticación única).
  - **Autenticación básica** (id. o clave de aplicación): una solución de cadena de **dos tokens**, es decir, **nombre de usuario** y **contraseña**.
  - OpenID Connect (OIDC): una capa de **identidad simple** sobre el **framework OAuth** conocido (es decir, verifica al usuario obteniendo información básica del perfil y utilizando un servidor de autenticación).
- Al seleccionar **un administrador de APIs**, debe saber **cuáles y cuántos de estos esquemas de seguridad** puede manejar, y **diseñar un plan** para que pueda incorporar las prácticas de seguridad de las APIs descritas anteriormente.