UNIVERSIDAD TECNOLOGICA DE SANTA CATARINA

APLICACIONES WEB ORIENTADA A SERVICIOS

TAREA 1 Y 2

ING. SEM TOMAS HERNANDEZ MARTINEZ

XIMENA MANZANAREZ RAMIREZ

JUAN ALBERTO LIÑAN MARTINEZ

BRANDON DANIEL ARCOS NUÑEZ

ROBERTO ALVAREZ HERNANDEZ

JUAN ANGEL HERNANDEZ JUAREZ

SANTA CATARINA, NUEVO LEÓN A 23 DE MAYO 2025

TAREA 1.

¿Qué es SOA?

El enfoque SOA (Arquitectura Orientada a Servicios) es una forma de desarrollar software basada en el uso de servicios independientes, los cuales se combinan para construir aplicaciones empresariales. Cada servicio proporciona una funcionalidad específica del negocio y puede interactuar con otros servicios, incluso si están construidos en distintas plataformas o lenguajes de programación. Los desarrolladores emplean esta arquitectura para aprovechar servicios ya existentes en varios sistemas o integrar múltiples servicios autónomos y así llevar a cabo procesos más complejos.

Por ejemplo, si en una organización varios procesos requieren autenticación de usuarios, no es necesario crear el mismo código cada vez. En su lugar, se puede desarrollar un servicio de autenticación único que sea reutilizable en todas las aplicaciones que lo necesiten. Del mismo modo, en el ámbito de la salud, distintos sistemas como los administrativos, de gestión de pacientes o de historiales médicos electrónicos pueden compartir un mismo servicio centralizado para registrar pacientes, evitando así la duplicación de funciones.

Principios de SOA.

Interoperabilidad

En la arquitectura SOA, cada servicio cuenta con documentación que detalla su funcionalidad y las condiciones para su uso. Esto permite que cualquier sistema cliente pueda acceder a los servicios, sin importar la plataforma o el lenguaje de programación con el que estén desarrollados. Por ejemplo, un proceso de negocio puede utilizar servicios escritos en distintos lenguajes como C# o Python. Al no existir una conexión directa entre los servicios, los cambios en uno no afectan a los demás componentes que lo emplean.

Acoplamiento

Los servicios en SOA deben diseñarse con un acoplamiento débil, es decir, con la menor dependencia posible de otros recursos externos como bases de datos o sistemas de información. Además, deben ser sin estado, lo que significa que no deben guardar información de sesiones o transacciones previas. Esto permite modificar un servicio sin que se vean perjudicadas las aplicaciones o servicios que lo utilizan.

Abstracción

Quienes consumen los servicios no necesitan conocer cómo están programados ni cómo funcionan internamente. Desde su perspectiva, los servicios funcionan como una

“caja negra”, y toda la información relevante sobre su uso se obtiene a través de contratos de servicio o documentos descriptivos.

Granularidad

Cada servicio en SOA debe estar diseñado con una función empresarial clara y específica. Los desarrolladores pueden combinar varios de estos servicios individuales para construir servicios más complejos que realicen tareas mayores.

¿Qué son los microservicios?

Los microservicios son un enfoque arquitectónico y organizativo para el desarrollo de software donde el software está compuesto por pequeños servicios independientes que se comunican a través de API bien definidas. Los propietarios de estos servicios son equipos pequeños independientes.

Las arquitecturas de microservicios hacen que las aplicaciones sean más fáciles de escalar y más rápidas de desarrollar. Esto permite la innovación y acelera el tiempo de comercialización de las nuevas características.

Características de los microservicios.

Autónomos

Cada servicio componente en una arquitectura de microservicios se puede desarrollar, implementar, operar y escalar sin afectar el funcionamiento de otros servicios. Los servicios no necesitan compartir ninguno de sus códigos o implementaciones con otros servicios. Cualquier comunicación entre componentes individuales ocurre a través de API bien definidas.

Especializados

Cada servicio está diseñado para un conjunto de capacidades y se enfoca en resolver un problema específico. Si los desarrolladores aportan más código a un servicio a lo largo del tiempo y el servicio se vuelve complejo, se puede dividir en servicios más pequeños.

Protocolos de red.

Los protocolos de red son un conjunto de reglas y normas que determinan cómo se transmiten y reciben datos a través de una red. Estos protocolos permiten que los

dispositivos se comuniquen entre sí, sin importar su marca, sistema operativo o ubicación.

Estándares de comunicación.

Los estándares de comunicación son especificaciones técnicas creadas por organizaciones internacionales para garantizar que los dispositivos y sistemas puedan trabajar entre sí sin problemas. Establecen cómo deben diseñarse y comportarse los componentes de red.

Los protocolos funcionan dentro de los estándares de comunicación. Por ejemplo, HTTP es un protocolo que se ajusta a las reglas del modelo OSI y utiliza TCP/IP como base para transmitir datos. Así, los estándares proporcionan el marco y los protocolos, las instrucciones precisas.

¿Qué es una API?

Las API son herramientas que facilitan la interacción entre dos programas de software, utilizando un conjunto de reglas y especificaciones. Por ejemplo, el sistema del instituto meteorológico almacena información sobre el clima cada día. La aplicación del clima en tu celular se conecta con ese sistema mediante API, lo que le permite mostrarte las condiciones meteorológicas actualizadas directamente en tu dispositivo.

API, o "interfaz de programación de aplicaciones", se refiere a un conjunto de reglas que permiten la interacción entre diferentes programas. En este contexto, una "aplicación" es cualquier software que realiza una función específica. La interfaz actúa como un acuerdo entre dos aplicaciones, estableciendo la forma en que intercambian información a través de mensajes de solicitud y respuesta. La documentación de la API ofrece las instrucciones necesarias para que los desarrolladores construyan correctamente esas comunicaciones.

TAREA 2.

¿Qué son las API REST?

REST significa transferencia de estado representacional. REST define un conjunto de funciones como GET, PUT, DELETE, etc. que los clientes pueden utilizar para acceder a los datos del servidor. Los clientes y los servidores intercambian datos mediante HTTP.

La principal característica de la API de REST es que no tiene estado. La ausencia de estado significa que los servidores no guardan los datos del cliente entre las solicitudes. Las solicitudes de los clientes al servidor son similares a las URL que se escriben en el navegador para visitar un sitio web. La respuesta del servidor son datos simples, sin la típica representación gráfica de una página web.

Las API de REST ofrecen cuatro beneficios principales:

1. Integración

Las API se utilizan para integrar nuevas aplicaciones con los sistemas de software existentes. Esto aumenta la velocidad de desarrollo, ya que no hay que escribir cada funcionalidad desde cero. Puede utilizar las API para aprovechar el código existente.

2. Innovación

Sectores enteros pueden cambiar con la llegada de una nueva aplicación. Las empresas deben responder con rapidez y respaldar la rápida implementación de servicios innovadores. Para ello, pueden hacer cambios en la API sin tener que reescribir todo el código.

3. Ampliación

Las API presentan una oportunidad única para que las empresas satisfagan las necesidades de sus clientes en diferentes plataformas. Por ejemplo, la API de mapas permite la integración de información de los mapas en sitios web, Android, iOS, etc. Cualquier empresa puede dar un acceso similar a sus bases de datos internas mediante el uso de API gratuitas o de pago.

4. Facilidad de mantenimiento

La API actúa como una puerta de enlace entre dos sistemas. Cada sistema está obligado a hacer cambios internos para que la API no se vea afectada. De este modo, cualquier cambio futuro que haga una de las partes en el código no afectará a la otra.

¿Cómo protegerla?

Todas las API deben protegerse mediante una autenticación y una supervisión adecuadas. Las dos maneras principales de proteger las API de REST son las siguientes:

1. Tokens de autenticación

Se utilizan para autorizar a los usuarios a hacer la llamada a la API. Los tokens de autenticación comprueban que los usuarios son quienes dicen ser y que tienen los derechos de acceso para esa llamada concreta a la API. Por ejemplo, cuando inicia sesión en el servidor de correo electrónico, el cliente de correo electrónico utiliza tokens de autenticación para un acceso seguro.

2. Claves de API

Las claves de API verifican el programa o la aplicación que hace la llamada a la API. Identifican la aplicación y se aseguran de que tiene los derechos de acceso necesarios para hacer la llamada a la API en cuestión. Las claves de API no son tan seguras como los tokens, pero permiten supervisar la API para recopilar datos sobre su uso. Es posible que haya notado una larga cadena de caracteres y números en la URL de su navegador cuando visita diferentes sitios web. Esta cadena es una clave de la API que el sitio web utiliza para hacer llamadas internas a la API.

Bibliografía

¿Qué es AOS? - Explicación sobre la arquitectura orientada a servicios - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/service-oriented-architecture/

¿Qué son los microservicios? | *AWS*. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/microservices/

IBM Sterling Order Management System. (s. f.). https://www.ibm.com/docs/es/order-management?topic=organization-defining-communication-protocols

¿Qué es una API? - Explicación de interfaz de programación de aplicaciones - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/api/

¿Qué es una API? - Explicación de interfaz de programación de aplicaciones - AWS. (s. f.-b). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/api/