Servidores de aplicaciones y middleware

Tema 4

Prof. Ing. Angel Brito Segura

1. Introducción

Los servidores de aplicaciones y middlewares desempeñan un papel crucial en la infraestructura de TI de las organizaciones modernas ya que permiten la integración eficiente de aplicaciones, bases de datos y servicios web, proporcionando un entorno robusto para el procesamiento de datos y la ejecución de aplicaciones distribuidas. Este tipo de servidores son los que soportan el desarrollo y la entrega de una aplicación de escritorio, móvil o web.

2. Servidores de Aplicaciones

Programas de servidor en una red distribuida que proporciona un entorno de ejecución para programas de aplicaciones. Más específicamente, este tipo de servidores son el componente de tiempo de ejecución principal en todas las configuraciones y donde se ejecuta una aplicación.

El servidor de aplicaciones colabora con el servidor web para devolver una respuesta dinámica y personalizada a una solicitud del cliente, por lo tanto es un marco mixto de software que permite tanto la creación de aplicaciones web como un entorno de servidor para ejecutarlas.

A menudo puede ser una pila compleja de diferentes elementos computacionales que ejecutan tareas específicas que necesitan trabajar como uno solo para alimentar múltiples nubes, software y aplicaciones basadas en la web.

Es habitual que se utilice junto con un servidor web o que contenga un servidor web, por lo que ambos pueden converger y denominarse servidor de aplicaciones web. También es lo suficientemente versátil como para ser utilizado con otros servidores de aplicaciones simultáneamente.

Los servidores de aplicaciones también pueden contener sus propias interfaces gráficas de usuario para su gestión, pero también pueden ocuparse de sus propios recursos, así como del procesamiento de transacciones, mensajería, agrupación de recursos y conexiones, así como la realización de tareas de seguridad.

Miles de millones de clientes web hacen peticiones HTTP cada día, esperando un acceso instantáneo a la aplicación en cuestión. Los servidores web se encargan de servir a los clientes web peticiones HTTP con respuestas HTTP. A diferencia de los servidores de aplicaciones, el diseño del servidor web es lo suficientemente ligero como para procesar las solicitudes de datos estáticos de varias aplicaciones (o sitios web),

manteniendo la seguridad.

Para conseguir una agilidad óptima del servidor web, no sirve gestionar tanto las peticiones HTTP de los clientes web como pasar o almacenar recursos de múltiples sitios web. Los servidores de aplicaciones llenan este vacío con un diseño de alta potencia construido para manejar las solicitudes de contenido web dinámico y proporcionan redundancia de programas con una capa adicional de seguridad.
Como la mayoría de los servidores de hoy en día, los servidores de aplicaciones contienen características de seguridad, transacciones, servicios, clustering, diagnósticos y bases de datos. En lo que se diferencian los servidores de aplicaciones es en su capacidad para procesar peticiones de servlets desde un servidor web.

2.1. Funcionamiento

El flujo general de los servidores de aplicaciones web son:

1. El cliente abre un navegador y solicita acceso a un sitio web.
2. El servidor web recibe la petición HTTP y responde con la página web deseada.
3. El servidor web gestiona las peticiones de datos estáticos, pero el cliente quiere utilizar una herramienta interactiva.
4. Al tratarse de una petición de datos dinámicos, el servidor web transfiere la petición a un servidor de aplicaciones.
5. El servidor de aplicaciones recibe la petición HTTP y la convierte en una petición para el servidor de la base de datos.
6. El servidor de aplicaciones recibe una respuesta de la base de datos.
7. El servidor de aplicaciones traduce la respuesta al formato HTTP para el acceso del cliente.

Dado que los servidores de aplicaciones trabajan principalmente con peticiones de lógica de negocio, el servidor web traduce la respuesta del servidor de aplicaciones y pasa una respuesta HTTP accesible para el usuario.

Los servidores de aplicaciones suelen ofrecer:

- Gestión de conexiones y sesiones.
- Seguridad y autenticación.
- Balanceo de carga y escalabilidad.
- Integración con bases de datos y servicios web.
- Soporte para múltiples lenguajes de programación y frameworks.
- Gestión de transacciones y monitoreo de rendimiento.

3. Middleware

Es una capa de superestructura de software por encima de la capa de transporte de comunicaciones de red que agrega servicios y abstracciones adicionales, actuando como un puente entre diferentes aplicaciones o sistemas, facilitando la comunicación y el intercambio de datos de manera transparente. Su propósito principal es proporcionar una capa de abstracción que simplifica la interoperabilidad entre aplicaciones heterogéneas.

El middleware agiliza el desarrollo de aplicaciones y acelera el tiempo de comercialización a través de funciones inteligentes que facilitan las conexiones entre plataformas que inicialmente no fueron diseñadas para conectarse, como por ejemplo:

- Agentes de mensajes o monitores de procesamiento de transacciones que se centran en un tipo de comunicación.
- Servidores de aplicaciones web o middlewares de dispositivos móviles que proporcionan toda la gama de capacidades de comunicación y conectividad necesarias para crear un tipo particular de aplicación.
- Plataforma de integración basada en la nube como servicio (iPaaS) o bus de servicio empresarial (EBS) que funciona como un hub de integración centralizado para conectar todos los componentes de una empresa.
- Middleware que permite a los desarrolladores construir su propio middleware personalizado.

3.1. Orígenes

Introducido por primera vez a finales de la década de 1960, el término middleware se acuñó porque la primera versión solía actuar como mediador entre el frontend de una aplicación (cliente) y un recurso de backend (por ejemplo, una base de datos, una aplicación de mainframe o un dispositivo de hardware especializado) desde la que el cliente podría aplicar datos. En respuesta al aumento de la computación distribuida en la década de 1980, el uso de middleware aumentó como una forma de vincular las aplicaciones más nuevas a los sistemas heredados tradicionales.

El middleware evolucionó para desempeñar un papel esencial en el desarrollo de aplicaciones modernas nativas de la nube. Emplea tecnología de contenedores, que permite la conexión a recursos distribuidos en entornos multinube. Esto soporta la integración continua (CI) y la entrega continua (CD), lo que agiliza la programación, las pruebas y el despliegue de aplicaciones para un rápido escalamiento y crecimiento del negocio.

3.2. Funcionamiento

El middleware permite a los desarrolladores construir aplicaciones sin crear una integración personalizada siempre que necesiten conectarse a componentes de aplicaciones (servicios o microservicios), fuentes de datos, recursos informáticos o dispositivos.

Para ello, el middleware proporciona servicios que permiten que diferentes aplicaciones y servicios se

comuniquen a través de marcos de mensajería comunes, como la notación de objetos JavaScript (JSON), la transferencia de estado representacional (REST), el lenguaje de marcado extensible (XML), el protocolo simple de acceso a objetos (SOAP) o los servicios web (HTTP). Normalmente, el middleware también proporciona servicios que permiten que los componentes escritos en varios lenguajes de programación (Java, C++, PHP y Python) se comuniquen entre sí.

3.3. Componentes

Los componentes arquitectónicos de un middleware estándar son:

1. ***Consola de administración de middleware***: ofrece a los desarrolladores una visión general de los eventos, actividades y configuraciones de middleware, etc.
2. ***Interfaz de cliente***: desde el frontend se comunica con aplicaciones backend, bases de datos, microservicios u otros servicios.
3. ***Interfaz interna de middleware***: permite a las instancias de middleware comunicarse entre sí con un protocolo especial de middleware.
4. ***Interfaz de la plataforma***: conecta a servidores backend y diferentes sistemas operativos, lo que garantiza las funciones de middleware en todas las plataformas.
5. ***Administrador de contratos***: define reglas de intercambio de datos que deben seguir las aplicaciones.
6. ***Administrador de sesiones***: garantiza una comunicación segura entre el middleware y las aplicaciones, y que no se agote el tiempo de espera.
7. ***Administrador de bases de datos***: proporciona integración con diferentes bases de datos en función de requisitos específicos (por ejemplo, datos en premisas o basados en la nube).
8. ***Monitor de tiempo de ejecución***: realiza un seguimiento de todos los movimientos de datos de middleware y proporciona informes de actividad para los desarrolladores de software.

3.4. Tipos

Existen muchos tipos de middleware: unos se enfocan en tipos específicos de conectividad, otros en aplicaciones particulares, componentes de aplicaciones y dispositivos. Algunos combinan capacidades de middleware para una tarea específica de desarrollo de software.

3.4.1. Middleware orientado a mensajes (MOM)

Este middleware permite que los componentes de la aplicación que emplean diferentes protocolos de mensajería se comuniquen e intercambien mensajes. Además de traducir o transformar mensajes entre aplicaciones, MOM gestiona el enrutamiento de los mensajes, por lo que siempre llegan a los componentes adecuados en el orden apropiado. Ejemplos de MOM incluyen colas de mensajes y agentes de mensajes como **RabbitMQ**.

3.4.2. Middleware de llamada a procedimiento remoto (RPC)

Este tipo de middleware permite a una aplicación activar un procedimiento en otra aplicación -que se ejecuta en la misma computadora o en otra computadora o red- como si ambas formaran parte de la misma aplicación en la misma computadora. Ejemplo de RPC es el desarrollado por Google llamado **gRPC** que se basa en Protocol Buffers (protobuf) como formato de serialización.

3.4.3. Middleware de datos o bases de datos

Este tipo de middleware simplifica el acceso y la interacción con las bases de datos de backend. Normalmente, el middleware de base de datos es algún tipo de servidor de base de datos SQL o NoSQL que se ejecuta en una red distribuida. Ejemplos de middleware de base de datos son **MySQL** y **MongoDB**.

3.4.4. Middleware de interfaz de programación de aplicaciones (API)

Este tipo de middleware proporciona herramientas que los desarrolladores pueden usar para crear, exponer y gestionar API para sus aplicaciones para que otros desarrolladores puedan conectarse a ellas.

Algunos middleware de API incluyen herramientas para monetizar las API, lo que permite que diferentes organizaciones las empleen, a un costo. Algunos ejemplos de middleware de API son las plataformas de gestión de API, las puertas de enlace de API y los portales para desarrolladores de API como lo es **WSO2 API Manager** y **Apigee**.

3.4.5. Middleware de Object Request Broker (ORB)

Este middleware actúa como intermediario entre una solicitud de un objeto o componente de la aplicación y el cumplimiento de esa solicitud por otro objeto o componente en la red distribuida. Ejemplo de este tipo de middleware son: DCOM, Jini, Java RMI y CORBA.

Los ORB funcionan con la arquitectura CORBA (Common Object Request Broker Architecture), que permite a un componente de software hacer una petición a otro sin saber dónde está alojado el otro ni cómo es su interfaz de usuario (UI): `brokering` gestiona esta información durante el intercambio.

3.4.6. Middleware transaccional

Este tipo de middleware proporciona servicios para respaldar la ejecución de transacciones de datos en una red distribuida. El middleware transaccional más conocido son los monitores de procesamiento de transacciones (TPM), que impulsan las transacciones de un paso al siguiente (ejecutando el intercambio de datos y agregando, cambiando o eliminando datos cuando sea necesario, entre otros) hasta su finalización.

3.4.7. Middleware de transmisión de datos asincrónicos

Este middleware replica un flujo de datos en un almacén intermedio, lo que permite el intercambio de datos entre múltiples aplicaciones. **Apache Kafka** es uno de los ejemplos más conocidos de plataformas de software medio de código abierto para la transmisión de eventos en tiempo real.

3.4.8. Middleware de dispositivo

Este tipo de middleware proporciona un conjunto enfocado de capacidades de integración y conectividad para desarrollar aplicaciones para un sistema operativo móvil específico. Ejemplos de middleware de dispositivos incluyen **Android Hardware Abstraction Layer (HAL)**, **Apple CoreMotion** y **Google Play Services**.

3.4.9. Middleware del portal

Este middleware proporciona herramientas y recursos para integrar contenidos y capacidades de varias aplicaciones relacionadas "en el cristal" o en una sola pantalla para crear una única aplicación compuesta. Ejemplos de este tipo de middleware son: **Liferay** y **IBM WebSphere Portal**.

3.4.10. Middleware de robótica

Este middleware simplifica la integración de hardware, firmware y software robóticos de múltiples fabricantes y ubicaciones. Ejemplos de middleware de robótica son **ROS** y **Microsoft Robotics Developer Studio**.

3.5. Categorías

Existen dos clases básicas de middleware:

3.5.1. Middleware de integración de aplicaciones empresariales

Estos middlewares permiten a una organización establecer un centro de integración empresarial. Esto proporciona una forma estandarizada de conectar todas las aplicaciones, componentes de aplicaciones, procesos de negocio y fuentes de datos de backend en toda la empresa extendida.

Hasta hace aproximadamente una década, el middleware de integración de aplicaciones empresariales más frecuente era el bus de servicios empresariales (ESB), que servía como centro de integración dentro de una arquitectura orientada a servicios (SOA).

Hoy en día, la plataforma de integración como servicio (iPaaS) permite a una organización conectar aplicaciones, datos, procesos y servicios a través de una nube híbrida: la combinación de entornos locales, de nube privada y de nube pública. Esto ayuda a las organizaciones a evitar el trabajo y los gastos de comprar, instalar, gestionar y mantener el middleware de integración (y el hardware en el que se ejecuta) dentro de su propio centro de datos. Los principales proveedores de servicios en la nube (CSP) ofrecen soluciones de este tipo.

3.5.2. Middleware de plataforma

El middleware de plataforma (o middleware de plataforma de aplicación) soporta el desarrollo de aplicaciones. Acelera la entrega de aplicaciones proporcionando un entorno de alojamiento en tiempo de ejecución -como un entorno en tiempo de ejecución Java (Java RE), contenedores o ambos- para la lógica de las aplicaciones o del negocio. Este middleware puede incluir o combinar servidores de aplicaciones empresariales, servidores web, sistemas de gestión de contenidos (CMS) y otros tipos de middleware.

3.6. Casos de uso

Además de proporcionar una interoperabilidad que ahorra trabajo, los servicios de middleware ayudan a los desarrolladores de software de las siguientes maneras:

3.6.1. Configurar y controlar conexiones e integraciones

Según la información de solicitud de la aplicación cliente o frontend, el middleware puede personalizar la respuesta de la aplicación o servicio de backend. Por ejemplo, en la aplicación de comercio electrónico de un minorista, la lógica de la aplicación de middleware puede ordenar los resultados de la búsqueda de productos de una base de datos de inventario de backend por la ubicación de almacenamiento más cercana en función de la dirección IP o la información de ubicación en el encabezado de la solicitud HTTP.

3.6.2. Proteger las conexiones y la transferencia de datos

El middleware generalmente establece una conexión segura de la aplicación frontend a las fuentes de datos backend que utilizan Transport Layer Security (TSL) u otro protocolo de seguridad de red. Puede proporcionar capacidades de autenticación, solicitudes desafiantes de aplicaciones frontend para credenciales (nombre de usuario y contraseña) o certificados digitales.

3.6.3. Gestionar el tráfico de forma dinámica en todos los sistemas distribuidos

Cuando el tráfico de aplicaciones aumenta, el middleware empresarial puede escalar para distribuir las solicitudes de los clientes en varios servidores, on-premises o en la nube. Las capacidades de procesamiento simultáneo pueden evitar problemas cuando varios clientes intentan acceder simultáneamente al mismo origen de datos backend.

3.6.4. Optimizar las aplicaciones existentes

El middleware ayuda a los desarrolladores a llevar a cabo la modernización de las aplicaciones, transformando las aplicaciones heredadas monolíticas en aplicaciones en la nube basadas en la arquitectura de microservicios.

3.6.5. Reducir las tareas manuales

La automatización de middleware ayuda a los desarrolladores a optimizar y automatizar la administración de tareas complejas de TI para mejorar la eficiencia general.

3.6.6. Admitir DevSecOps

El middleware respalda las metodologías DevSecOps (que significa desarrollo, seguridad y operaciones) al ayudar a los equipos a crear aplicaciones más rápido y, al mismo tiempo, mitigar los riesgos de seguridad.

3.6.7. Desarrollo nativo en la nube

Nativo de la nube es un enfoque de desarrollo de aplicaciones que usa tecnologías fundamentales de computación en la nube para proporcionar un desarrollo, despliegue y gestión congruentes en entornos de

nube híbrida.

Las aplicaciones nativas de la nube de hoy en día se crean a partir de microservicios y se implementan en contenedores que emplean Kubernetes, una plataforma de orquestación de contenedores ampliamente empleada.

Los microservicios son componentes de aplicaciones débilmente acoplados que abarcan su propia pila y se pueden implementar y actualizar de forma independiente entre sí. Se comunican mediante una combinación de API REST, agentes de mensajes y flujos de eventos.

Los contenedores son tareas de ejecución ligeras que empaquetan código de aplicación, como microservicios, junto con las bibliotecas del sistema operativo y las dependencias necesarias para ejecutar ese código en cualquier nube o infraestructura de TI tradicional.

Juntas, estas y otras tecnologías relacionadas crean una poderosa plataforma de desarrollo único y despliegue en cualquier lugar para ofrecer aplicaciones de nube híbrida totalmente nuevas y modernizar los sistemas heredados tradicionales para su uso en la nube. Sin embargo, también conducen a un entorno de desarrollo complejo que combina aún más aplicaciones de software, fuentes de datos, lenguajes de programación, herramientas y sistemas distribuidos.

El middleware puede resolver parte de esta complejidad. Sin embargo, la ejecución de aplicaciones en contenedores con middleware convencional puede agregar complejidades propias, incluida la sobrecarga de infraestructura para la que se diseñaron los contenedores. Por esta razón, las plataformas de desarrollo de aplicaciones en la nube incluyen middleware en contenedores o modulares para que solo las funciones de conectividad requeridas se puedan empaquetar en un contenedor.

3.6.8. Desarrollo de videojuegos

El middleware respalda el desarrollo de juegos al servir como motor de juego. Esta capa de software ayuda a integrar sin problemas video, audio y otros componentes cruciales del juego.

3.6.9. Servicios financieros

Para las instituciones financieras, el middleware integra aplicaciones y bases de datos de clientes para dar soporte a los servicios bancarios digitales, incluido el procesamiento de transacciones en tiempo real.

3.6.10. Atención médica

Siempre fue difícil acceder a los datos de salud y compartirlos de forma segura. El middleware es crucial para la interoperabilidad de la atención médica, ya que permite un flujo de datos fluido en varios sistemas y aplicaciones de atención médica, como plataformas de telemedicina e historias clínicas electrónicas (EHR).

3.6.11. Comercio electrónico

Las organizaciones de comercio electrónico emplean middleware para conectar sus plataformas en línea con servicios back-end cruciales, como el procesamiento de pedidos, para mejorar la experiencia general del cliente.

3.6.12. Manufactura

En la fabricación, el middleware ayuda a garantizar que los sistemas de software empresarial, incluidas las plataformas de planeación de recursos empresariales (ERP) , se integren con otras aplicaciones para proporcionar una visión unificada de las operaciones. Esto ayuda a optimizar y agilizar el mantenimiento, la cadena de suministro, el control de calidad y más.

4. Balanceo de Carga

El balanceo de carga es esencial para distribuir las solicitudes de los usuarios entre múltiples servidores, evitando sobrecargas y mejorando la disponibilidad del sistema.

4.1. Técnicas de Balanceo de Carga

Existen varias estrategias para distribuir la carga entre servidores:

- **Round Robin**: Distribuye solicitudes de manera equitativa entre los servidores disponibles.
- **Least Connections**: Redirige las solicitudes al servidor con menos conexiones activas.
- **IP Hashing**: Asigna un servidor a cada cliente según su dirección IP.
- **Balanceo basado en peso**: Permite asignar más solicitudes a servidores con mayor capacidad.

4.2. Herramientas de Balanceo de Carga

Las soluciones más utilizadas incluyen:

- **NGINX**: Software ampliamente utilizado para el balanceo de carga y la gestión de tráfico web.
- **HAProxy**: Solución de código abierto que ofrece balanceo de carga y alta disponibilidad.

Referencias

[1] M. Burgess, *Principles of Network and System Administration*. John Wiley & Sons, 2004.

[2] J. Bergstra y M. Burgess, *Handbook of Network and System Administration*. Elsevier, 2007.

[3] Raúl. «Servidores de aplicaciones.» (2022), dirección: https://raul-profesor.github.io/Despliegue/ServAplic/. (accedido: 05.03.2025).

[4] S. Susnjara e I. Smalley, *¿Qué es el middleware?* IBM, 2024. dirección: https://www.ibm.com/mx-es/topics/middleware, (accedido: 05/03/2025).