Sistemas Distribuidos

Bloque II

Desarrollo de sistemas distribuidos y aplicaciones web.

Tema 3 .2

Tecnologías de comunicación de aplicaciones

URJC Escuela Técnica Superior de Ingeniería Informática

Nicolás Rodríguez
nicolas.rodriguez@urjc.es

Índice

Formatos de datos: JSON y XML

- Los formatos de datos son esenciales para el intercambio de información entre diferentes sistemas y aplicaciones.
- JSON (JavaScript Object Notation):
  - Formato ligero de intercambio de datos.
  - Fácil de leer y escribir para humanos, y fácil de analizar y generar para máquinas.
  - Basado en el subconjunto de JavaScript, pero independiente del lenguaje: se usa en muchos lenguajes de programación.
- XML (eXtensible Markup Language):
  - Formato que define un conjunto de reglas para codificar documentos de forma legible tanto para máquinas como para humanos.
  - Altamente personalizable y extensible.
  - A menudo utilizado en aplicaciones que requieren un alto grado de complejidad y estructura de datos, como en la industria de software empresarial.

Formatos de datos: JSON y XML

- Comparación y uso:
  - Verbosidad: XML es más detallado y extenso, mientras que JSON es más compacto y eficiente en términos de tamaño.
  - Facilidad de Uso: JSON es generalmente más fácil de usar y rápido en el procesamiento, ideal para aplicaciones web y móviles.
  - Estructura de Datos: XML es más adecuado para documentos con una estructura compleja y datos jerárquicos.
- Ejemplos de aplicaciones:
  - JSON en APIs web, configuraciones de aplicaciones y almacenamiento de datos simples.
  - XML en servicios web, documentos de Office y configuraciones de software más complejas.

Índice

Protocolos de red: TCP/IP, HTTP/HTTPS/SMTP

- Los protocolos de red son un conjunto de reglas que permiten la comunicación entre dispositivos en una red.
- Son fundamentales para el intercambio de datos en Internet y en redes locales.
- TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet):
  - Es un conjunto de protocolos de comunicación que interconectan dispositivos en Internet.
  - TCP se encarga de la entrega confiable de datos, mientras que IP se ocupa de la dirección y enrutamiento de paquetes.
  - Importancia: Esencial para la mayoría de las formas de comunicación en Internet, como la navegación web y el envío de correos electrónicos.

Protocolos de red: TCP/IP, HTTP/HTTPS/SMTP

- HTTP/HTTPS (Protocolo de Transferencia de Hipertexto/Protocolo Seguro):
  - HTTP es el protocolo utilizado para transferir datos en la World Wide Web.
  - HTTPS es la versión segura de HTTP, encriptando la comunicación para proteger los datos intercambiados.
  - Importancia: HTTP es fundamental para la navegación web, y HTTPS ha aumentado en importancia debido a la necesidad de seguridad y privacidad en línea.
- Funcionamiento y aplicaciones:
  - TCP/IP funciona dividiendo los mensajes en paquetes y ensamblándolos en el destino.
  - HTTP/HTTPS se utiliza en cada solicitud y respuesta en la web, desde cargar páginas hasta realizar transacciones en línea.

Protocolos de red: TCP/IP, HTTP/HTTPS/SMTP

- Simple Mail Transfer Protocol es un protocolo estándar para el envío de correos electrónicos en Internet:
  - Envío de correo electrónico: SMTP se utiliza principalmente para enviar mensajes de correo electrónico desde clientes de correo electrónico a servidores y entre servidores de correo.
  - Protocolo basado en texto: Utiliza comandos de texto simples para la comunicación, lo que facilita su implementación y depuración.
  - Proceso de entrega: En una transacción SMTP típica, un cliente se conecta al servidor SMTP del remitente, el servidor autentica al remitente (si es necesario), y luego el servidor transfiere el mensaje al servidor SMTP del destinatario.
  - Relay y routing: SMTP se encarga de dirigir y retransmitir los mensajes a través de una serie de servidores de correo hasta que el mensaje llega a su destino.

Índice

- Formatos de datos (JSON, XML)
- Protocolos de red (TCP/IP, HTTP/HTTPS, SMTP)
- **Servicios Web (SOAP y REST)**
- Protocolos de mensajería (MQTT, AMQP)
- Middleware y colas de mensajes
- Apache Kafka
- RabbitMQ
- AWS SQS

Servicios Web: SOAP y REST

- Los servicios web son interfaces que permiten la interacción entre aplicaciones a través de la red. Utilizan un conjunto de protocolos y estándares para asegurar que distintas máquinas puedan comunicarse entre sí.
- SOAP (Protocolo Simple de Acceso a Objetos):
  - Basado en XML para el intercambio de información.
  - Utiliza principalmente HTTP y SMTP para la comunicación.
  - Orientado a acciones con un enfoque en la funcionalidad y métodos.
- REST (Transferencia de Estado Representacional):
  - No depende de un protocolo específico, pero comúnmente usa HTTP.
  - Basado en la arquitectura de recursos y servicios web.
  - Utiliza métodos HTTP estándar como GET, POST, PUT y DELETE.

Servicios Web: SOAP y REST

- Comparación clave:
  - Flexibilidad: REST es generalmente más flexible y fácil de usar que SOAP.
  - Seguridad: SOAP tiene un estándar de seguridad más robusto (WS-Security).
  - Peso de los Datos: Las solicitudes REST son más ligeras en comparación con SOAP, mejorando la velocidad y eficiencia.
- Ejemplos de uso:
  - SOAP es comúnmente usado en sistemas bancarios y financieros por su seguridad y formalidad en las transacciones.
  - REST se utiliza ampliamente en aplicaciones web y móviles debido a su simplicidad y eficiencia.

Índice

- Formatos de datos (JSON, XML)
- Protocolos de red (TCP/IP, HTTP/HTTPS, SMTP)
- Servicios Web (SOAP y REST)
- **Protocolos de mensajería (MQTT, AMQP)**
- Middleware y colas de mensajes
- Apache Kafka
- RabbitMQ
- AWS SQS

Protocolos de Mensajería: MQTT y AMQP

- Son conjuntos de reglas que permiten el intercambio eficiente de mensajes entre sistemas y aplicaciones.
- Juegan un papel crucial en la Internet de las Cosas (IoT) y en la comunicación entre sistemas distribuidos.

Protocolos de Mensajería: MQTT y AMQP

- MQTT (Message Queuing Telemetry Transport):
  - Diseñado para conexiones con ancho de banda limitado y alta latencia.
  - Utiliza el modelo de publicación/suscripción, siendo ideal para dispositivos IoT.
  - Ofrece tres niveles de calidad de servicio (QoS), garantizando así la entrega de mensajes bajo diferentes condiciones de red.
- AMQP (Advanced Message Queuing Protocol):
  - Protocolo de mensajería orientado a mensajes, robusto y seguro.
  - Permite una amplia variedad de interacciones de mensajería, incluyendo enrutamiento de mensajes, transacciones y colas.
  - Adaptable para sistemas empresariales complejos y para garantizar la fiabilidad y seguridad en la entrega de mensajes.

Protocolos de Mensajería: MQTT y AMQP

- Diferencias clave:
  - Escalabilidad: MQTT es más adecuado para dispositivos y redes con recursos limitados, mientras que AMQP está diseñado para sistemas con necesidades más complejas.
  - Modelo de Mensajería: MQTT se centra en el modelo de publicación/suscripción, mientras que AMQP ofrece una gama más amplia de patrones de mensajería.
- Ejemplos de uso:
  - MQTT en dispositivos domésticos inteligentes, vehículos autónomos y apps de salud.
  - AMQP en sistemas financieros, logística y comunicaciones internas de empresas.

Índice

Middleware y colas de mensajes

- El middleware es un software que proporciona servicios comunes y capacidades de comunicación entre aplicaciones y componentes de software.
- Actúa como un puente entre diferentes aplicaciones y bases de datos.
- Funciones clave del Middleware:
  - Facilita la comunicación y el manejo de datos entre aplicaciones distribuidas.
  - Proporciona servicios como autenticación, autorización, y gestión de sesiones.
  - Ofrece funcionalidades como balanceo de carga y manejo de transacciones.

Middleware y colas de mensajes

- Colas de mensajes:
  - Son componentes de middleware que ayudan a gestionar y distribuir mensajes entre diferentes aplicaciones.
  - Permiten la comunicación asíncrona, donde el emisor y el receptor no necesitan estar en línea al mismo tiempo.
  - Ayudan a desacoplar procesos en SSDD, aumentando la escalabilidad y la resiliencia.
- Ejemplos de Middleware y colas de mensajes:
  - Middleware: Oracle Middleware, IBM WebSphere.
  - Colas de Mensajes: Apache Kafka, RabbitMQ, AWS SQS.
- Ventajas del uso de Middleware y colas de mensajes:
  - Mejora la interoperabilidad entre diferentes sistemas y tecnologías.
  - Facilita la escalabilidad y la gestión de grandes volúmenes de datos y transacciones.
  - Aumenta la fiabilidad y la disponibilidad de los sistemas distribuidos.

Colas de mensajes

- Las colas de mensajes son componentes fundamentales en la arquitectura de sistemas distribuidos.
- Permiten la comunicación asíncrona entre diferentes partes de un sistema, donde los productores envían mensajes y los consumidores los reciben y procesan.
- Cómo funcionan las colas de mensajes:
  - Los mensajes se almacenan en una cola hasta que pueden ser procesados.
  - Aseguran que los mensajes se entreguen y procesen en el orden en que se recibieron.
  - Proporcionan un mecanismo para manejar picos de carga, distribuyendo los mensajes a medida que los sistemas están disponibles para procesarlos.

Colas de mensajes

- Características importantes:
  - Fiabilidad: Garantizan que los mensajes no se pierdan, incluso en caso de fallos en el sistema.
  - Escalabilidad: Facilitan el manejo de grandes volúmenes de mensajes y la expansión de sistemas.
  - Desacoplamiento: Los productores y consumidores operan independientemente, mejorando la modularidad y mantenibilidad.
- Ejemplos y herramientas Populares:
  - Apache Kafka: Orientado a alto rendimiento y distribución.
  - RabbitMQ: Ampliamente usado, enfocado en la simplicidad y facilidad de uso.
  - AWS SQS (Simple Queue Service): Solución en la nube, integrable con otros servicios de AWS.

Índice

Apache Kafka

- Kafka es una plataforma de código abierto diseñada para procesar y manejar streams de datos en tiempo real.
- Fue desarrollado inicialmente por LinkedIn y luego donado a la Apache Software Foundation.
- Características principales:
  - Alto Rendimiento y Escalabilidad: Puede manejar miles de mensajes por segundo, escalando horizontalmente para soportar grandes volúmenes de datos.
  - Modelo de Publicación/Suscripción: Los productores envían mensajes a topics (temas), y los consumidores los leen, facilitando una comunicación efectiva y eficiente.
  - Durabilidad y Fiabilidad: Almacena los mensajes en discos, lo que garantiza que no se pierdan incluso en caso de fallos del sistema.

Apache Kafka

- Arquitectura de Kafka:
  - Brokers: Servidores que almacenan los datos y sirven a los consumidores.
  - ZooKeeper: Utilizado para la coordinación y gestión de los brokers de Kafka.
  - Producers y Consumers: Aplicaciones que publican y leen mensajes desde los topics.
- Casos de uso comunes:
  - Procesamiento de streams en tiempo real: Análisis de datos en vivo, como monitoreo de transacciones financieras.
  - Sistemas de recomendación: Procesamiento de actividades de usuarios para generar recomendaciones personalizadas.
  - Integración de datos y microservicios: Como una capa de comunicación entre diferentes microservicios en una arquitectura empresarial.

Índice

- Formatos de datos (JSON, XML)
- Protocolos de red (TCP/IP, HTTP/HTTPS, SMTP)
- Servicios Web (SOAP y REST)
- Protocolos de mensajería (MQTT, AMQP)
- Middleware y colas de mensajes
- Apache Kafka
- **RabbitMQ**
- AWS SQS

RabbitMQ

- RabbitMQ es un sistema de mensajería de código abierto que actúa como un intermediario de mensajes (message broker) para aplicaciones.

- Es ampliamente utilizado para la comunicación entre diferentes componentes de un sistema, permitiendo la transmisión de mensajes de manera confiable y eficiente.
- RabbitMQ implementa AMQPaunque también soporta otros protocolos como MQTT, STOMP y HTTP.

RabbitMQ – Conceptos clave

- Colas de mensajes: RabbitMQ permite a las aplicaciones enviar y recibir mensajes a través de colas, que son almacenamientos de mensajes hasta que son procesados.
- Publicación y suscripción: Los productores publican mensajes en colas y los consumidores los suscriben y procesan.
- Modelo de intercambio (Exchange): Controla cómo se enrutan los mensajes entre las colas. Existen varios tipos de intercambio, como directo, tópico, encabezados y fanout.

RabbitMQ – Conceptos clave

- Confiabilidad: Garantiza la entrega de mensajes a través de funcionalidades como la confirmación de mensajes y la persistencia.
- Escalabilidad y alto rendimiento: Puede manejar un gran volumen de mensajes y es altamente escalable.
- Clustering y tolerancia a fallos: Soporta clustering para mayor disponibilidad y resistencia a fallos.
- Soporte para múltiples lenguajes y plataformas: Puede ser utilizado con una variedad de lenguajes de programación a través de bibliotecas cliente.

RabbitMQ – Aplicaciones

- Desacoplamiento de aplicaciones: Permite que diferentes componentes de una aplicación se comuniquen de manera asincrónica, aumentando la modularidad.
- SSDD: Facilita la comunicación en sistemas distribuidos, mejorando la escalabilidad y la eficiencia.
- Manejo de cargas de trabajo: Utilizado para distribuir tareas entre varios trabajadores (workers) y balancear la carga.
- Integraciones de sistemas: Facilita la integración de sistemas y aplicaciones diferentes.

Índice

- Formatos de datos (JSON, XML)
- Protocolos de red (TCP/IP, HTTP/HTTPS, SMTP)
- Servicios Web (SOAP y REST)
- Protocolos de mensajería (MQTT, AMQP)
- Middleware y colas de mensajes
- Apache Kafka
- RabbitMQ
- **AWS SQS**

AWS SQS

- AWS SQS (Amazon Web Services Simple Queue Service) es un servicio de colas de mensajes completamente gestionado que permite la desvinculación y el escalado de microservicios, sistemas distribuidos y aplicaciones sin servidor.
- Ofrece una solución robusta y escalable para el manejo de mensajes entre componentes de software en la nube de AWS.

amazon
SQS

AWS SQS – Conceptos clave

- SQS almacena mensajes en múltiples servidores para asegurar su persistencia, ofreciendo durabilidad a largo plazo (hasta 14 días).
- Con SQS, los usuarios pueden enviar, almacenar y recibir un número ilimitado de mensajes entre componentes de software en cualquier momento, sin pérdida de mensajes.
- Dos tipos de colas:
  - Colas Estándar: Ofrecen throughput máximo, entrega al menos una vez y un orden de mensajes que puede no ser exacto.
  - Colas FIFO (First-In-First-Out): Aseguran que los mensajes se entreguen y procesen exactamente una vez y en el orden exacto en que se envían.

AWS SQS – Conceptos clave

- Escalabilidad: SQS escala automáticamente con la aplicación, por lo que no es necesario administrar la infraestructura de mensajería.
- Seguridad: Los usuarios pueden utilizar IAM (Identity and Access Management) para controlar el acceso a las colas SQS y crifrar mensajes para asegurar datos sensibles.
- Integración: SQS se integra con otros servicios de AWS, así como con sistemas de notificaciones y BBDD, lo que facilita la orquestación de flujos de trabajo y el procesamiento de datos.

AWS SQS – Aplicaciones

- Desacoplamiento de aplicaciones: Permite que las diferentes partes de una aplicación se comuniquen sin estar conectadas directamente, mejorando la fiabilidad y la flexibilidad.
- Manejo de cargas de trabajo: SQS puede ser utilizado para suavizar picos de tráfico almacenando mensajes hasta que los sistemas estén listos para procesarlos.
- Orquestación de flujos de trabajo: Al integrarse con otros servicios de AWS, SQS se puede utilizar para orquestar flujos de trabajo complejos y procesamiento de transacciones.

Sistemas Distribuidos

Bloque II

Desarrollo de sistemas distribuidos y aplicaciones web.

Tema 3 .2

Tecnologías de comunicación de aplicaciones

URJC Escuela Técnica Superior de Ingeniería Informática

Nicolás Rodríguez
nicolas.rodriguez@urjc.es