UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
CAMPUS 1

ING. EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE

6 “M”

TALLER DE DESARROLLO 4

SUBCOMPETENCIA III

ACT. 3.1 INVESTIGAR SISTEMAS DE INFORMACIÓN CON ARQUITECTURAS BASADAS EN MICROSERVICIOS

ALUMNO: MARCO ANTONIO ZÚÑIGA MORALES – A211121

DOCENTE: DR. LUIS GUTIÉRREZ ALFARO

TUXTLA GUTIÉRREZ, CHIAPAS
JUEVES, 26 DE OCTUBRE DE 2023

Índice

Introducción

En un mundo cada vez más digitalizado, las arquitecturas basadas en microservicios han surgido como una metodología fundamental en el diseño y desarrollo de sistemas de información. La arquitectura de microservicios se basa en la idea de dividir una aplicación en componentes independientes y autónomos, llamados microservicios, que realizan tareas específicas y se comunican entre sí a través de API (Interfaz de Programación de Aplicaciones). Esto permite una mayor agilidad, escalabilidad y mantenibilidad de las aplicaciones, al tiempo que promueve la reutilización de código y la distribución eficiente de recursos.

La arquitectura de microservicios ha revolucionado la forma en que desarrollamos, integramos y gestionamos aplicaciones en la era digital. En esta investigación, exploraremos en detalle diversos aspectos relacionados con los microservicios, desde sus conceptos fundamentales hasta ejemplos concretos de su implementación en algunas de las empresas más influyentes del mundo, como Netflix, Amazon Web Services, Facebook, Twitter y Spotify. Comenzaremos analizando las características esenciales de los microservicios y los diferentes tipos que existen, así como los requisitos funcionales y no funcionales que los guían. Además, exploraremos las ventajas que ofrecen los microservicios en la integración de aplicaciones y su papel en el diseño de sistemas y las pruebas de software. A través de casos de estudio, desglosaremos la relación entre Netflix y la arquitectura de microservicios, destacando las motivaciones detrás de su migración, las características clave de su arquitectura y las ventajas resultantes. Asimismo, analizaremos cómo Netflix optimiza la experiencia de streaming utilizando la ciencia de datos y resuelve desafíos de big data. Finalmente, exploraremos cómo otras empresas líderes como Amazon Web Services, Facebook, Twitter y Spotify implementan microservicios en sus arquitecturas y los beneficios que obtienen de este enfoque innovador. Esta investigación proporcionará una visión completa de los microservicios y su influencia en el mundo de la tecnología y el desarrollo de aplicaciones.

Desarrollo

Microservicio

Los microservicios son tanto un estilo de arquitectura como un modo de programar software. Con los microservicios, las aplicaciones se dividen en sus elementos más pequeños e independientes entre sí. A diferencia del enfoque tradicional y monolítico de las aplicaciones, en el que todo se compila en una sola pieza, los microservicios son elementos independientes que funcionan en conjunto para llevar a cabo las mismas tareas. Cada uno de esos elementos o procesos es un microservicio.

Los sistemas de información basados en arquitecturas de microservicios son una tendencia en el diseño de software que se ha vuelto cada vez más popular en los últimos años. Esta arquitectura se caracteriza por dividir una aplicación en componentes independientes, llamados microservicios, que realizan tareas específicas y se comunican entre sí a través de API (Interfaz de Programación de Aplicaciones). Cada microservicio es autocontenido y puede ser desarrollado, implementado y escalado de forma independiente.

Características de un microservicio

- **Componentes:**
  Los microservicios suelen estar formados por componentes de software discretos que son actualizables y reemplazables de forma individual. Esta arquitectura tiene implicaciones para la tecnología de administración de la nube porque cada uno de los microservicios debe estar aprovisionado, monitoreado y actualizado por separado.

- **Servicios:**
  Los componentes comprenden servicios que están disponibles para comunicarse a pedido, pero pueden no estar activos de forma continua entre las solicitudes o las llamadas.

- **Implementación independiente:**
  En su mayor parte, los componentes de servicios individuales operan de forma independiente uno de otro dentro del marco de microservicios. Si se cambia o actualiza un componente, hay poco impacto en otros servicios y componentes, especialmente cuando se compara con una arquitectura monolítica más tradicional.

- **Seguridad:**
  La comunicación entre los microservicios suele estar cifrada con seguridad de la capa de transporte mutuo (mTLS) para proteger los datos del malware y las intrusiones mientras está en tránsito.

- **Contenerización:**
  Los microservicios suelen implementarse en contenedores para lograr escalabilidad y portabilidad adicionales.

¿Qué tipos de microservicio existen?

Los microservicios se pueden dividir en cinco diferentes categorías: API, Procesamiento de datos, Almacenamiento, Lógica de negocio e Interfaz de usuario.

- **Api**
  Un servicio API es una capa de interfaz para el intercambio de información entre aplicaciones y servicios. Esto le permite a un determinado servicio o aplicación exponer una interfaz única para usuarios externos e internos para una variedad de funcionalidades.

  Un ejemplo de API es un servicio que expone una interfaz para la obtención de recomendaciones de productos. Esto le permite a una aplicación obtener información de productos específicos y exponer información de los mismos al usuario final.

- **Procesamiento de datos**
  Los microservicios de procesamiento de datos le permiten a una aplicación recopilar, procesar y almacenar grandes cantidades de datos. Estos procesos se realizan en pequeños paquetes de cálculo y se compilan a un gran volumen de datos.

  Un ejemplo es un servicio que procesa datos en tiempo real y los almacena en una base de datos. Esto le da a una aplicación la capacidad de procesar datos a gran escala sin necesidad de una infraestructura de procesamiento de datos masiva.

- **Almacenamiento**
  Los microservicios de almacenamiento le permiten a una aplicación almacenar y recuperar datos de una base de datos centralizada o una gran base de datos distribuida. Esto le da a a la aplicación la capacidad de procesar y almacenar grandes volúmenes de datos sin necesidad de una gran infraestructura de base de datos.

- **Lógica de negocios**
  Los microservicios de lógica de negocio son servicios que se encargan de realizar tareas específicas relacionadas con el funcionamiento del negocio. Esto incluye la validación de pedidos, transacciones financieras, gestión de inventario y otras tareas relacionadas.

  Un ejemplo de microservicios de lógica de negocio es un servicio que realiza un seguimiento de las transacciones realizadas por un usuario. Esto le da a

una aplicación la capacidad de procesar las transacciones de los usuarios de manera segura y eficiente.

- **Interfaz de usuario**

  Los microservicios de interfaz de usuario le permiten a una aplicación servir interfaces de usuario optimizadas para diferentes dispositivos a la vez. Esto incluye pantallas sensibles al tamaño, diseños adaptables a la orientación y diseños optimizados para distintos sistemas operativos.

  Un ejemplo es un servicio que personaliza la interfaz de usuario de una aplicación en función de los datos y preferencias del usuario. Esto le da a una aplicación la capacidad de ofrecer una experiencia de usuario mejorada de manera individual.

Requisitos funcionales y no funcionales

Los requisitos se pueden clasificar en funcionales y no funcionales. Los requisitos funcionales (RF) describen la funcionalidad que los usuarios esperan del sistema. Los requisitos no funcionales (RNF) son restricciones impuestas sobre el sistema a desarrollar, estableciendo por ejemplo como de rápido o fiable ha de ser. Mientras que los primeros no incluyen ninguna mención relacionada con la tecnología que emplea el sistema, los segundos sí pueden establecer restricciones de este tipo.

Por ejemplo, un requisito no funcional puede consistir en desarrollar una aplicación en un lenguaje de programación específico o hacer que esté disponible para diferentes sistemas operativos móviles. Por este motivo, los requisitos deben ser tenidos en cuenta a lo largo de todo el desarrollo del sistema.

Características y subcaracterísticas definidas en el modelo de calidad del producto de la ISO/IEC 25010

Ventajas de los microservicios en la integración de las aplicaciones

- **Agilidad**
  Dado que los microservicios se implementan de forma independiente, resulta más fácil administrar las correcciones de errores y las versiones de características, ya que el servicio se puede actualizar sin volver a implementar toda la aplicación y revertir una actualización si algo va mal. En muchas aplicaciones tradicionales, un error en una parte de la aplicación puede bloquear todo el proceso de lanzamiento, por lo que es posible que se requieran nuevas características a la espera de que se integre, pruebe y publique una corrección de errores.

- **Equipos pequeños y centrados**
  Un microservicio debe ser lo suficientemente pequeño como para que un solo equipo lo pueda compilar, probar e implementar. Mientras los equipos pequeños favorecen la agilidad, los equipos grandes suelen ser menos productivos porque la comunicación es más lenta, aumenta la sobrecarga de administración y la agilidad disminuye.

- **Base de código pequeña**
  En las aplicaciones monolíticas, con el paso del tiempo se da la tendencia de que las dependencias del código se acaban enredando. Para agregar una característica nueva, es preciso modificar el código en muchos lugares.

- **Aislamiento de errores**
  Si un microservicio individual no está disponible, no interrumpirá toda la aplicación, siempre que los microservicios ascendentes estén diseñados para controlar los errores correctamente. Por ejemplo, puede implementar o diseñar la solución para que los microservicios se comuniquen entre sí mediante patrones de mensajería asincrónica.

- **Escalabilidad**
  Los servicios se pueden escalar de forma independiente, lo que permite escalar horizontalmente los subsistemas que requieren más recursos, sin tener que escalar horizontalmente toda la aplicación. Mediante un orquestador como Kubernetes o Service Fabric se puede empaquetar una mayor densidad de servicios en un solo host, lo que aumenta la eficacia en el uso de los recursos.

- **Aislamiento de los datos**
  Al verse afectado solo un microservicio, es mucho más fácil realizar actualizaciones del esquema. En una aplicación monolítica, las actualizaciones del esquema pueden ser muy complicadas, ya que las distintas partes de la aplicación pueden tocar los mismos datos, por lo que realizar modificaciones en el esquema resulta peligroso.

Los microservicios en el diseño de sistema

En la actividad de diseño se definen la arquitectura, componentes e interfaces del sistema. La especificación de requisitos es analizada para producir una descripción de la estructura interna del sistema, con el suficiente nivel de detalle para que sirva como base en su construcción.

- **Librerías**
  Las librerías son componentes que están ligados a un programa y son invocadas a través de llamadas a funciones

- **Servicios**
  Los servicios son componentes que se ejecutan como procesos externos y con los que se puede comunicar a través de mecanismos como llamadas a procedimientos remotos (RPC) o peticiones HTTP.

- **Diseño guiado por el dominio (DDD)**
  Es un enfoque para el desarrollo de software que propone un modelado rico, expresivo y evolutivo basado en la realidad del negocio.

Los microservicios en la actividad de pruebas

Las pruebas de software consisten en la verificación de que un programa produce las salidas esperadas para un conjunto finito de casos de prueba. Los casos de prueba son finitos porque el número posible de pruebas es infinito y estos se seleccionan en función de la prioridad y riesgo del código bajo pruebas.

- **Pruebas unitarias**
  Una prueba unitaria es una pieza de código que invoca al método o clase bajo prueba y que comprueba ciertas asunciones sobre su lógica. Se ejecutan de forma rápida y sencilla, están automatizados y son fáciles de mantener.

- **Pruebas de servicios**
  En las pruebas de servicios se verifican cada una de las funcionalidades que un servicio expone. Se pretende verificar el servicio de forma aislada y para ignorar las dependencias que el servicio bajo pruebas tiene sobre otros se reemplazan los servicios colaboradores por fakes.

- **Pruebas de extremo a extremo**
  Las pruebas de extremo a extremo son pruebas que se ejecutan sobre todo el sistema. Cubren gran parte de código, por lo que su correcta ejecución da un alto grado de confianza.

Ejemplos de sistemas de información basados en arquitectura de microservicios:

- Netflix
- Amazon Web Service (AWS)
- Facebook
- Twitter
- Spotify
- Airbnb
- Uber

Caso de estudio: Netflix

Netflix es un servicio de transmisión en línea que ofrece una amplia variedad de películas y programas de televisión a millones de usuarios en todo el mundo.

Netflix es un conocido servicio de transmisión de video en línea que utiliza una arquitectura basada en microservicios para brindar contenido de video a millones de usuarios en todo el mundo.

¿Por qué Microservicios está asociado con Netflix?

Netflix es una de las primeras empresas en migrar con éxito de una arquitectura monolítica tradicional a una arquitectura de microservicios basadas en la nube. De hecho, Netflix implementó esta arquitectura mucho antes de que se introdujera el término microservicios. Netflix tardó más de dos años en lograr la migración completa a la nube. Netflix no solo perfeccionó el uso de microservicios, sino que también logró abrir muchas de las herramientas que se usaron para construirlo. El OSS de Netflix (Centro de software de código abierto) tiene muchas herramientas y tecnologías que otras empresas pueden utilizar para crear una arquitectura de microservicios en la nube.

Motivo de esta migración

Cuando Netflix anunció su gran cambio a la nube, se enfrentaron a muchas críticas, ya que nadie creía que tal hazaña fuera posible en ese momento. La razón principal por la que Netflix decidió pasar a la nube se debió al rápido aumento de los datos y la información del usuario que era difícil de almacenar en sus centros de datos actuales, lo que causó una gran cantidad de problemas. La solución se logró utilizando Amazon Web Service (AWS), que prometía proporcionar grandes recursos informáticos y centros de datos con seguridad y confiabilidad garantizadas. Con AWS, el escalado se puede realizar en un par de minutos sin la participación del usuario.

Mientras se trasladaba a la nube, Netflix logró dividir su única aplicación monolítica en cientos de pequeños servicios poco acoplados. Hoy, Netflix tiene más de 1000 microservicios, cada uno de los cuales administra una parte separada del sitio.

Características clave de su arquitectura

1. **Gestión de usuarios y autenticación:**
   Netflix maneja millones de cuentas de usuario. Los microservicios relacionados con la gestión de usuarios se encargan de la autenticación, la autorización y la administración de perfiles. Cada uno de estos servicios es independiente y puede escalar según la demanda.

2. **Catalogación de contenido:**
   La plataforma de Netflix contiene una amplia variedad de contenido, desde películas hasta series y documentales. Cada tipo de contenido y sus metadatos se gestionan a través de microservicios separados. Esto permite una escalabilidad eficiente y una administración centralizada de la información del catálogo.

3. **Reproducción de video:**
   La reproducción de video es esencial en Netflix. Los microservicios relacionados con la transmisión de video se encargan de la entrega de contenido multimedia a través de Internet. Esto incluye la administración de la calidad del video, la selección de pistas de audio y subtítulos, y la adaptación de la velocidad de bits según la conexión del usuario.

4. **Sugerencias y recomendaciones:**
   Los algoritmos de recomendación de Netflix son impulsados por microservicios que analizan el historial de visualización, las calificaciones y las preferencias de los usuarios para ofrecer sugerencias personalizadas. Estos servicios trabajan en segundo plano y proporcionan una experiencia de usuario más atractiva.

5. **Monitoreo y escalabilidad:**
   Netflix utiliza herramientas de monitoreo y administración para garantizar que los microservicios funcionen sin problemas. La plataforma puede escalar automáticamente la capacidad de los microservicios según la demanda, lo que es crucial para mantener un rendimiento constante durante los picos de uso.

6. **Tecnologías utilizadas:**
   Netflix utiliza una variedad de tecnologías, incluyendo Java, Spring Boot, y otras herramientas de código abierto para implementar y administrar sus microservicios. También hacen uso de plataformas en la nube, como AWS (Amazon Web Services), para alojar y escalar su infraestructura.

7. **Facturación y pagos:**
   La gestión de pagos y facturación se maneja a través de microservicios separados, lo como facturación, procesamiento de tarjetas de crédito y administración de suscripciones.

Ventajas de la arquitectura de microservicios de Netflix

- **Escalabilidad:**
  Cada microservicio puede escalarse de forma independiente, lo que permite gestionar cargas de trabajo variables de manera eficiente.

- **Mantenimiento sencillo:**
  Los equipos pueden trabajar de forma independiente en microservicios, lo que facilita la implementación de nuevas características y correcciones de errores sin afectar a todo el sistema.

- **Tiempo de comercialización más rápido:**
  La arquitectura de microservicios permite a Netflix implementar nuevas características y mejoras de manera más rápida, lo que mantiene a los usuarios comprometidos.

- **Mejora de la disponibilidad:**
  Si un microservicio falla, no afecta a todo el sistema. Netflix puede aislar y solucionar problemas de manera eficiente sin que afecte a la experiencia del usuario en su conjunto.

Pruebas de Netflix como microservicio

1. **Pruebas unitarias:**
   Cada microservicio debe tener pruebas unitarias sólidas que se centren en las funciones y lógica específica del servicio. Estas pruebas pueden incluir pruebas de casos normales y casos límite para garantizar la robustez del microservicio.

2. **Pruebas de integración:**
   Se deben realizar pruebas de integración para garantizar que los microservicios funcionen juntos de manera adecuada. Esto implica verificar la comunicación y la interoperabilidad entre los servicios.

3. **Pruebas de rendimiento:**
   Netflix maneja un gran volumen de tráfico, por lo que las pruebas de rendimiento son esenciales. Esto incluye pruebas de carga para verificar cómo los microservicios responden bajo cargas de trabajo pesadas. También es importante realizar pruebas de estrés para determinar los límites de capacidad de los servicios.

4. **Pruebas de seguridad:**
   La seguridad es crítica en una plataforma como Netflix. Se deben realizar pruebas de seguridad para identificar posibles vulnerabilidades, como la inyección de SQL, la exposición de datos sensibles o la autenticación defectuosa.

5. **Pruebas de recuperación ante fallos:**
   Los microservicios deben ser resistentes a fallos. Las pruebas de recuperación ante fallos son esenciales para garantizar que un servicio pueda recuperarse de manera adecuada en caso de una interrupción.

6. **Pruebas de monitoreo y registro:**
   Netflix utiliza herramientas de monitoreo y registro para supervisar el rendimiento de sus microservicios. Las pruebas de monitoreo deben garantizar que se están recopilando los datos adecuados y que se pueden identificar problemas rápidamente.

7. **Pruebas de actualización y despliegue:**
   Las actualizaciones de los microservicios deben ser probadas exhaustivamente para garantizar que no causen interrupciones en el servicio. Esto puede incluir pruebas de canary deployment y blue-green deployment.

8. **Pruebas de escalabilidad:**
   Dado que los microservicios deben ser escalables, es importante realizar pruebas para garantizar que un servicio pueda escalarse horizontalmente de manera efectiva.

9. **Pruebas de tolerancia a fallos:**
   Netflix utiliza el caos engineering para simular fallos en su plataforma y garantizar que los microservicios sean tolerantes a fallos. Estas pruebas pueden incluir la introducción deliberada de fallos en el sistema para evaluar cómo responde.

10. **Pruebas de regresión:**
    Cada vez que se realiza una actualización en un microservicio, se deben ejecutar pruebas de regresión para garantizar que las nuevas modificaciones no hayan afectado la funcionalidad existente.

Servicios de Netflix como microservicio

- **Catálogo de Contenido (Content Catalog Microservice):**
  Este microservicio manejaría la gestión y búsqueda de contenido, incluyendo películas y programas de televisión. Proporcionaría información sobre títulos, actores, directores, géneros y más.
- **Reproducción de Video (Video Playback Microservice):**

Este servicio se encargaría de la entrega de video en streaming. Gestionaría la reproducción de video, la calidad del video y la administración de listas de reproducción.

- **Recomendaciones (Recommendations Microservice):**
  Sería responsable de proporcionar recomendaciones personalizadas para los usuarios, basadas en su historial de visualización y preferencias.

- **Perfiles de Usuario (User Profiles Microservice):**
  Este microservicio administraría los perfiles de usuario, incluyendo la creación, edición y eliminación de perfiles. También gestionaría las preferencias de idioma y subtítulos.

- **Sistema de Autenticación (Authentication Microservice):**
  Sería responsable de gestionar la autenticación y autorización de los usuarios, incluyendo el inicio de sesión, la administración de contraseñas y los tokens de seguridad.

- **Facturación y Suscripciones (Billing and Subscriptions Microservice):**
  Manejaría el procesamiento de pagos, la gestión de suscripciones, la facturación y la información de la tarjeta de crédito.

- **Estadísticas y Análisis (Analytics Microservice):**
  Este microservicio recopilaría datos de uso, como las estadísticas de visualización, la retención de usuarios y otros datos relevantes para el análisis.

- **Notificaciones (Notifications Microservice):**
  Sería responsable de enviar notificaciones a los usuarios, como recordatorios de nuevos contenidos o actualizaciones de sus programas favoritos.

- **Comentarios y Calificaciones (Comments and Ratings Microservice):**
  Gestionaría la capacidad de los usuarios para dejar comentarios y calificaciones en películas y programas de televisión.

- **Configuración y Personalización (Settings and Customization Microservice):**
  Administraría las configuraciones de la cuenta del usuario, como la calidad de video, las preferencias de subtítulos y la administración de dispositivos.
- **Servicio de Búsqueda (Search Service Microservice):**
  Proporcionaría capacidades de búsqueda avanzada, incluyendo la búsqueda de contenido, actores y directores.

- **Gestión de Permisos (Permissions Management Microservice):**
  Se encargaría de la gestión de permisos y roles de usuario para garantizar un acceso adecuado y seguro a los servicios de Netflix.

Optimización de la experiencia de streaming de Netflix con ciencia de datos – Por Nirmal Govind

Nirmal Govind (2014) nos dice, El 16 de enero de 2007, Netflix comenzó a implementar una nueva función: los miembros ahora podían transmitir películas directamente en su navegador sin tener que esperar el sobre rojo en el correo.
Este evento marcó un cambio sustancial para Netflix y la industria del entretenimiento. Mucho ha cambiado desde entonces. En la actualidad, Netflix ofrece más de 1.48 millones de horas de streaming al mes a 40 millones de miembros en más de países.

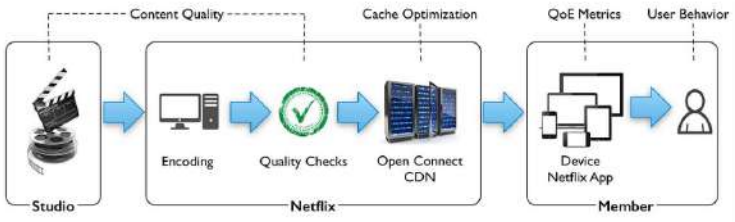
Big data

En Netflix, utilizamos big data para realizar análisis profundos y algoritmos predictivos que ayuden a brindar la mejor experiencia a nuestros miembros. Un ejemplo bien conocido de esto son las recomendaciones personalizadas de películas y programas que se adaptan a los gustos de cada miembro. El premio de Netflix que se lanzó en 2007 destacó el enfoque de Netflix en las recomendaciones. Otra área en la que nos estamos enfocando es la calidad de la experiencia de transmisión (QoE) que se refiere a la experiencia del usuario una vez que el miembro presiona reproducir en Netflix. Esta es un área que se beneficia significativamente de la ciencia de datos y los algoritmos/modelos construidos en torno al big data.

Tipos de problemas que resolvió Neftlix

- Comprender el impacto de la QoE en el comportamiento del usuario
- Creación de una experiencia de streaming personalizada para cada miembro
- Determinar qué películas y programas almacenar en caché en los servidores perimetrales en función del comportamiento de visualización de los miembros
- Mejorar la calidad técnica del contenido de nuestro catálogo utilizando los datos de visualización y los comentarios de los miembros.

Mejorar la experiencia de streaming



La cadena de suministro de streaming de Netflix: las oportunidades para optimizar la experiencia de streaming existen en múltiples puntos

Creación de una experiencia de streaming personalizada

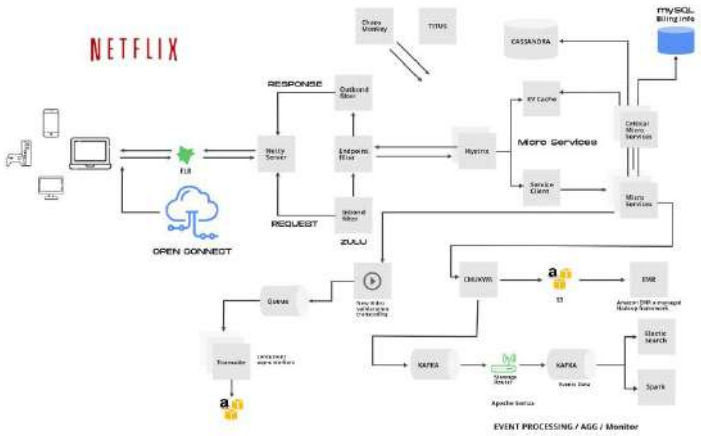
Un enfoque consiste en observar los algoritmos que se ejecutan en tiempo real o casi en tiempo real una vez que se ha iniciado la reproducción, que determinan qué tasa de bits se debe servir, desde qué servidor descargar ese contenido, etc.

Optimización del almacenamiento en caché de contenido

También existe un conjunto de problemas de big data en el lado de la entrega de contenido. Open Connect es la red de entrega de contenido propia de Netflix que permite a los ISP conectarse directamente a los servidores de Netflix en los intercambios de Internet comunes, o colocar un dispositivo de almacenamiento (caché) proporcionado por Netflix con contenido de Netflix en las ubicaciones de los ISP. La idea clave aquí es ubicar el contenido más cerca (en términos de saltos de red) de nuestros miembros para brindar una gran experiencia.

Arquitectura del sistema de alto nivel de Netflix

Netflix funciona en dos nubes… AWS y Open Connect . Estas dos nubes funcionan juntas como la columna vertebral de Netflix y ambas son muy responsables de brindar el mejor video a los suscriptores.



La aplicación tiene principalmente 3 componentes:

- **Cliente:**
  Dispositivo (interfaz de usuario) que se utiliza para buscar y reproducir videos de Netflix. TV, XBOX, ordenador portátil o teléfono móvil, etc.

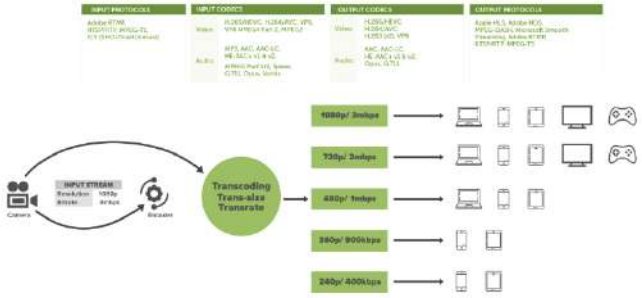
- **OC (Open connect) o Netflix CDN:**
  CDN es la red de servidores distribuidos en diferentes ubicaciones geográficas, y Open Connect es la propia CDN (red de entrega de contenido)

global personalizada de Netflix. Maneja todo lo que implica transmisión de video. Se distribuye en diferentes ubicaciones y una vez que presiona el botón de reproducción, la transmisión de video de este componente se muestra en su dispositivo. Entonces, si está tratando de reproducir el video desde Norteamérica, el video se entregará desde la conexión abierta (o servidor) más cercana en lugar del servidor original (respuesta más rápida del servidor más cercano).

- **Backend (base de datos):**
  Esta parte maneja todo lo que no implica la transmisión de video (antes de presionar el botón de reproducción), como incorporar contenido nuevo, procesar videos, distribuirlos a servidores ubicados en diferentes partes del mundo y administrar el tráfico de red. Amazon Web Services se encarga de la mayoría de los procesos.

¿Cómo incorpora Netflix una película/video?

Netflix recibe videos y contenido de muy alta calidad de las productoras, por lo que antes de entregar los videos a los usuarios, realiza un preprocesamiento. Netflix admite más de 2200 dispositivos y cada uno de ellos requiere resoluciones y formatos diferentes. Para que los videos se puedan ver en diferentes dispositivos, Netflix realiza la transcodificación o codificación, lo que implica encontrar errores y convertir el video original en diferentes formatos y resoluciones.

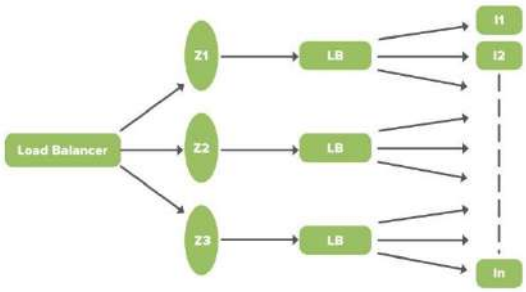


Los datos del usuario se guardan en AWS, como búsquedas, visualización, ubicación, dispositivo, reseñas y Me gusta, Netflix los usa para crear la recomendación de películas para los usuarios que usan el modelo de aprendizaje automático o Hadoop.

Ventajas de la conexión abierta

- Menos costoso
- Mejor calidad
- Más escalable

Equilibrador de carga elástico de Netflix



ELB en Netflix es responsable de enrutar el tráfico a los servicios frontend. ELB realiza un esquema de equilibrio de carga de dos niveles en el que la carga se equilibra primero entre las zonas y luego entre las instancias (servidores).

- El primer nivel consiste en un balanceo Round Robin básico basado en DNS. Cuando la solicitud llega al primer balanceo de carga (ver la figura), se balancea en una de las zonas (usando round-robin) que su ELB está configurado para usar.

- El segundo nivel es una array de instancias de equilibrador de carga y realiza la técnica de equilibrio por turnos para distribuir la solicitud entre las instancias que están detrás de ella en la misma zona.

ZUUL

ZUUL es un servicio de puerta de enlace que proporciona enrutamiento dinámico, supervisión, resiliencia y seguridad. Proporciona un enrutamiento fácil basado en parámetros de consulta, URL y ruta. Entendamos el funcionamiento de sus diferentes partes:

- **El servidor Netty** asume la responsabilidad de manejar el protocolo de red, el servidor web, la gestión de conexiones y el trabajo de proxy. Cuando la solicitud llegue al servidor de Netty, enviará la solicitud al filtro de entrada.

- **El filtro de entrada** es responsable de la autenticación, el enrutamiento o la decoración de la solicitud. Luego reenvía la solicitud al filtro de punto final.

- **El filtro de punto final** se usa para devolver una respuesta estática o para reenviar la solicitud al servicio de backend (u origen, como lo llamamos). Una vez que recibe la respuesta del servicio de backend, envía la solicitud al filtro de salida.

- **El filtro de salida** se usa para comprimir el contenido, calcular las métricas o agregar/eliminar encabezados personalizados. Después de eso, la respuesta se envía de vuelta al servidor de Netty y luego el cliente la recibe.

Ventajas

- Puede crear algunas reglas y compartir el tráfico distribuyendo las diferentes partes del tráfico a diferentes servidores.

- Los desarrolladores también pueden realizar pruebas de carga en clústeres recién implementados en algunas máquinas. Pueden enrutar parte del tráfico existente en estos clústeres y verificar cuánta carga puede soportar un servidor específico.

- También puede probar nuevos servicios . Cuando actualiza el servicio y desea verificar cómo se comporta con las requests de API en tiempo real, en ese caso, puede implementar el servicio en particular en un servidor y puede redirigir una parte del tráfico al nuevo servicio para verificar la servicio en tiempo real.

- También podemos filtrar la solicitud incorrecta configurando las reglas personalizadas en el filtro de punto final o en el firewall.

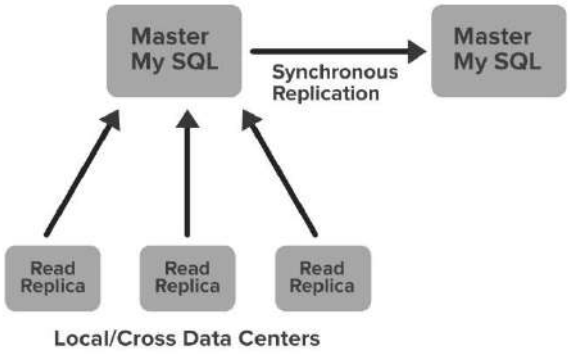
Base de datos de Netflix

Netflix utiliza dos bases de datos diferentes, es decir, MySQL (RDBMS) y Cassandra (NoSQL) para diferentes propósitos.

MySQL implementado en EC2

Netflix guarda datos como información de facturación, información de usuario e información de transacciones en MySQL porque necesita el cumplimiento de ACID. Netflix tiene una configuración maestro-maestro para MySQL y se implementa en grandes instancias EC2 de Amazon usando InnoDB.

La configuración sigue el Protocolo de replicación síncrona, donde si el escritor pasa al Node principal principal, también se replicará en otro Node principal. El reconocimiento se enviará solo si se ha confirmado la escritura de los Nodes maestro primario y remoto. Esto asegura la alta disponibilidad de los datos.



Casandra

Cassandra es una base de datos NoSQL que puede manejar grandes cantidades de datos y también puede manejar escritura y lectura pesadas. Cuando Netflix comenzó a adquirir más usuarios, los datos del historial de visualización de cada miembro también comenzaron a aumentar. Esto aumenta la cantidad total de datos del historial de visualización y se vuelve un desafío para Netflix manejar esta enorme cantidad de datos.

- Huella de almacenamiento más pequeña.
- Rendimiento constante de lectura/escritura a medida que crece la visualización por miembro (la relación de lectura/escritura de datos del historial de visualización es de aproximadamente 9:1 en Cassandra).

Modelo de datos desnormalizados totales

- Más de 50 clústeres de Cassandra
- Más de 500 Nodes
- Más de 30 TB de copias de seguridad diarias
- Clúster más grande 72 Nodes.
- 1 clúster de más de 250 000 escrituras/s

Caso de estudio: Amazon Web Service (AWS)

AWS es una plataforma de computación en la nube que ofrece una amplia gama de servicios, desde infraestructura básica hasta aplicaciones y análisis. AWS se basa en una arquitectura de microservicios, lo que le permite ofrecer una amplia gama de servicios de forma escalable, flexible y adaptable.

Características de su arquitectura

1. **Autenticación y autorización:**
   AWS utiliza microservicios dedicados para la autenticación y autorización de usuarios y aplicaciones que acceden a sus servicios. Estos microservicios gestionan la seguridad y el acceso a la infraestructura de AWS, garantizando que solo los usuarios autorizados puedan utilizar los recursos.

2. **Almacenamiento y bases de datos:**
   AWS ofrece una amplia gama de servicios de almacenamiento y bases de datos, como Amazon S3 (Simple Storage Service) y Amazon RDS (Relational Database Service). Cada uno de estos servicios se implementa como un microservicio independiente que se puede escalar según la demanda y se integra en la arquitectura de microservicios de AWS.

3. **Procesamiento y cómputo:**
   AWS proporciona servicios de cómputo en la nube, como Amazon EC2 (Elastic Compute Cloud). La gestión de estas instancias de cómputo se realiza a través de microservicios que permiten la creación, administración y escalabilidad de máquinas virtuales y contenedores de forma independiente.

4. **Redes y enrutamiento:**
   La infraestructura de red de AWS se basa en microservicios que gestionan la asignación de direcciones IP, la configuración de reglas de seguridad y el enrutamiento de tráfico entre instancias y servicios. Esto garantiza la escalabilidad y la seguridad de la red en la nube de AWS.

5. **Almacenamiento de objetos y CDN:**
   AWS ofrece servicios como Amazon S3 y Amazon CloudFront para el almacenamiento de objetos y la entrega de contenido a través de una red de distribución de contenido (CDN). Estos servicios se implementan como microservicios separados y se escalan según la demanda.

6. **Orquestación y administración de contenedores:**
   AWS proporciona servicios de orquestación de contenedores como Amazon ECS (Elastic Container Service) y Amazon EKS (Elastic Kubernetes Service). Estos servicios utilizan microservicios para administrar y orquestar contenedores de manera eficiente.

7. **Monitoreo y administración de recursos:**
   AWS utiliza microservicios para monitorear y administrar los recursos en la nube. AWS CloudWatch es un ejemplo de un servicio de monitoreo que recopila métricas y registros de los servicios y recursos en AWS.

8. **Escalabilidad y alta disponibilidad:**
   AWS está diseñado para ser altamente escalable y disponible. Esto se logra mediante la implementación de microservicios que pueden adaptarse a las demandas cambiantes y la distribución de cargas de trabajo en diferentes regiones y zonas de disponibilidad para garantizar la redundancia y la recuperación de desastres.

9. **Tecnologías utilizadas:**
   AWS utiliza diversas tecnologías y herramientas para implementar y gestionar sus microservicios, incluyendo tecnologías de contenedores como Docker y Kubernetes, así como herramientas de orquestación y automatización.

Beneficios de utilizar AWS para el desarrollo de microservicios

AWS ofrece una serie de beneficios para el desarrollo de microservicios, entre los que destacan:

- **Amplia gama de servicios:**
  AWS ofrece una amplia gama de servicios que pueden utilizarse para crear microservicios.

- **Escalabilidad:**
  AWS permite escalar los microservicios de forma independiente, lo que facilita la adaptación a las necesidades cambiantes del negocio.

- **Flexibilidad:**
  AWS permite desarrollar, desplegar y actualizar los microservicios de forma independiente, lo que facilita el mantenimiento y la evolución de las aplicaciones.

- **Reutilización:**
  AWS permite reutilizar los microservicios en diferentes aplicaciones, lo que reduce el coste de desarrollo.

Caso de estudio: Facebook

Facebook es una red social líder a nivel mundial que proporciona una amplia gama de servicios a sus usuarios. Este caso de estudio se centrará en cómo Facebook podría beneficiarse de la adopción de una arquitectura de microservicios para aumentar la escalabilidad, la flexibilidad y la eficiencia de su plataforma.

Características de su arquitectura

1. **Gestión de usuarios:**
   Facebook gestiona cientos de millones de cuentas de usuario. Los microservicios relacionados con la gestión de usuarios se encargan de la autenticación, la autorización, la gestión de perfiles, la administración de amigos y las preferencias de privacidad. Cada uno de estos microservicios es independiente y puede escalar según la demanda.

2. **Alimentación de noticias (News Feed):**
   El News Feed es una parte fundamental de la experiencia de usuario en Facebook. Los microservicios relacionados con el News Feed recopilan, filtran y entregan contenido a los usuarios en tiempo real. Esto incluye la clasificación de publicaciones, la gestión de publicidad y la detección de contenido no deseado.

3. **Chat y mensajería:**
   Facebook Messenger es una aplicación de mensajería independiente que se integra con la plataforma de Facebook. Los microservicios de chat y mensajería gestionan la entrega de mensajes, llamadas de voz y video, y la sincronización de chats en múltiples dispositivos.

4. **Gestión de medios y almacenamiento de fotos:**
   Facebook almacena y comparte una gran cantidad de contenido multimedia. Los microservicios de gestión de medios se encargan de la carga, el almacenamiento y la entrega de fotos y videos. Además, la generación de miniaturas y el reconocimiento de objetos en imágenes son tareas que se pueden realizar a través de microservicios.

5. **Análisis de datos y publicidad:**
   Los microservicios relacionados con la publicidad en Facebook se encargan de la orientación de anuncios, la administración de campañas publicitarias y la recopilación de datos de usuario para mejorar la segmentación. También pueden ofrecer análisis de rendimiento para anunciantes.

6. **Integración de aplicaciones y servicios externos:**
   Facebook permite a los desarrolladores de aplicaciones de terceros integrarse con su plataforma. Los microservicios de integración gestionan la

autenticación de aplicaciones externas, el acceso a datos de usuario y la interacción con API de terceros.

7. **Seguridad y privacidad:**
   Facebook debe abordar cuestiones de seguridad y privacidad en su plataforma. Los microservicios relacionados con la seguridad se encargan de la detección y mitigación de amenazas, así como de garantizar la privacidad de los datos del usuario.

8. **Escalabilidad y alta disponibilidad:**
   La arquitectura de microservicios permite a Facebook escalar y administrar la infraestructura de manera eficiente, asegurando que la plataforma pueda manejar millones de usuarios simultáneos y permanecer disponible en todo momento.

9. **Tecnologías utilizadas:**
   Facebook utiliza una variedad de tecnologías, incluyendo lenguajes de programación como PHP, Hack, y React, así como sistemas de almacenamiento y bases de datos personalizados para admitir su arquitectura de microservicios.

Caso de estudio: Twitter / X

Twitter, una de las plataformas de redes sociales más influyentes y ampliamente utilizadas en el mundo, utiliza una arquitectura basada en microservicios para respaldar su plataforma.

Características de su arquitectura

1. **Gestión de usuarios y autenticación:**
   Twitter maneja una gran cantidad de cuentas de usuario. Los microservicios relacionados con la gestión de usuarios se ocupan de la autenticación, la autorización y la administración de perfiles. Cada uno de estos microservicios es independiente y escalable, lo que permite a los usuarios acceder y administrar sus cuentas de manera eficiente.

2. **Línea de tiempo y publicación de tweets:**
   La funcionalidad central de Twitter implica la visualización de una línea de tiempo personalizada y la publicación de tweets. Los microservicios relacionados con la línea de tiempo se encargan de recopilar y organizar tweets de usuarios seguidos, mientras que los de publicación gestionan la creación y distribución de tweets a través de la plataforma.

3. **Búsqueda y tendencias:**
   Twitter utiliza microservicios para proporcionar funciones de búsqueda, incluyendo la búsqueda de tweets y usuarios. Además, los microservicios de tendencias rastrean los temas populares y los hashtags en tiempo real.

4. **Notificaciones y mensajes directos**:
   Los usuarios de Twitter pueden recibir notificaciones y mensajes directos. Los microservicios relacionados con notificaciones y mensajería gestionan la entrega de mensajes y alertas de manera eficiente.

5. **Gestión de medios y visualización de imágenes y videos:**
   Twitter permite a los usuarios compartir imágenes y videos. Los microservicios de gestión de medios se encargan de la carga, el almacenamiento y la entrega de medios compartidos, así como la generación de miniaturas y la visualización de contenido multimedia.

6. **Autenticación y seguridad:**
   La arquitectura de microservicios de Twitter se encarga de la autenticación de usuarios, la seguridad de la plataforma y la protección contra amenazas, como el spam y el abuso.

7. **Escalabilidad y alta disponibilidad:**
   Twitter necesita ser altamente escalable y disponible para manejar un gran volumen de tweets y usuarios en todo momento. Su arquitectura de microservicios permite una fácil escalabilidad y distribución en múltiples ubicaciones geográficas.

8. **Tecnologías utilizadas:**
   Twitter utiliza una variedad de tecnologías y herramientas, incluyendo Java, Scala, Ruby on Rails y tecnologías de bases de datos NoSQL, para implementar y gestionar sus microservicios. También hacen uso de sistemas de contenedores y orquestación para garantizar la disponibilidad y escalabilidad.

Caso de estudio: Spotify

Spotify, uno de los servicios de transmisión de música más populares del mundo, se basa en una arquitectura de microservicios para ofrecer una experiencia de usuario altamente personalizada y escalable.

Características de su arquitectura

1. **Gestión de usuarios y autenticación:**
   Spotify gestiona millones de cuentas de usuario. Los microservicios relacionados con la gestión de usuarios se encargan de la autenticación, la

autorización, la administración de perfiles y las preferencias de música de los usuarios. Cada uno de estos microservicios es independiente y escalable.

2. **Búsqueda y recomendaciones de música:**
   Spotify utiliza algoritmos de recomendación para ofrecer listas de reproducción y recomendaciones personalizadas. Los microservicios relacionados con la búsqueda y recomendación analizan el historial de escucha y las preferencias de los usuarios para proporcionar recomendaciones precisas y actualizadas.

3. **Streaming de música:**
   La reproducción de música en tiempo real es el núcleo del servicio de Spotify. Los microservicios relacionados con el streaming de música se encargan de la transmisión de canciones, la administración de la calidad del audio y la adaptación de la velocidad de bits según la conexión del usuario.

4. **Gestión de listas de reproducción y colecciones de música:**
   Los usuarios de Spotify pueden crear listas de reproducción y colecciones de canciones. Los microservicios de gestión de listas de reproducción permiten a los usuarios organizar y compartir su música de manera eficiente.

5. **Sincronización de dispositivos:**
   Spotify permite a los usuarios escuchar música en múltiples dispositivos. Los microservicios de sincronización garantizan una experiencia continua al permitir que los usuarios cambien de un dispositivo a otro sin interrupciones.

6. **Análisis de datos y publicidad:**
   Spotify utiliza microservicios para la recopilación y el análisis de datos de usuarios, lo que le permite personalizar aún más la experiencia de los usuarios y ofrecer publicidad dirigida. También ofrece una plataforma para anunciantes que pueden aprovechar la audiencia específica de Spotify.

7. **Monitoreo y escalabilidad:**
   Spotify utiliza herramientas de monitoreo y administración para garantizar que sus microservicios funcionen sin problemas. La plataforma puede escalar automáticamente la capacidad de los microservicios según la demanda, lo que es fundamental para mantener un rendimiento constante durante los picos de uso.

8. **Tecnologías utilizadas:**
   Spotify utiliza tecnologías como Java, Python, y Scala para implementar y administrar sus microservicios. También hacen uso de sistemas de contenedores y orquestación de contenedores para garantizar la disponibilidad y la escalabilidad.

Conclusión

En conclusión, de acuerdo con la información de libros, sitios web y artículos, podemos concluir que los microservicios han emergido como una arquitectura de desarrollo altamente efectiva y versátil que se ha convertido en la columna vertebral de numerosas organizaciones líderes, incluyendo gigantes tecnológicos como Netflix, Amazon Web Service (AWS), Facebook, Twitter y Spotify. Estos microservicios se caracterizan por su capacidad para abordar tanto los requisitos funcionales como no funcionales de manera eficiente, permitiendo una integración más fluida de aplicaciones, mejorando la escalabilidad, facilitando la prueba y optimización, y brindando ventajas significativas en la entrega de servicios y experiencias personalizadas a los usuarios. La arquitectura de microservicios se ha convertido en un motor clave para la innovación y el éxito en el mundo de la tecnología, impulsando mejoras en la experiencia de streaming, optimización del almacenamiento en caché, equilibrio de carga elástico y el uso de servicios en la nube como AWS. Las organizaciones líderes han adoptado esta arquitectura para mantenerse competitivas en un entorno en constante evolución, demostrando su capacidad para revolucionar la forma en que se desarrollan, entregan y mantienen aplicaciones y servicios.

Referencia Bibliográfica

Blog, N. T. (2018, 15 mayo). Optimizing the Netflix streaming experience with data science. Medium. https://netflixtechblog.com/optimizing-the-netflix-streaming-experience-with-data-science-725f04c3e834

Bravet, R. M. (s. f.). Arquitectura de microservicios con Spring y Netflix OSS aplicando prácticas DevOps. https://es.linkedin.com/pulse/arquitectura-de-microservicios-con-spring-y-netflix-oss-rene-martinez

¿Cómo se distribuye la arquitectura de microservicios de Netflix? – Programador Clic. (s. f.). https://programmerclick.com/article/99941660578/

Greyrat, R. (2022, julio 5). La historia de Netflix y los microservicios – Barcelona Geeks. https://barcelonageeks.com/la-historia-de-netflix-y-los-microservicios/

¿Qué son y para qué sirven los microservicios? Beneficios de los microservicios. (s. f.). https://www.redhat.com/es/topics/microservices

Whitestack. (2023, 29 agosto). Microservicios: ¿Qué son y qué tipos existen? ¿Cómo funciona su arquitectura? https://whitestack.com/es/blog/microservicios/

Netflix Technology Blog. (n.d.). Recuperado de https://netflixtechblog.com/

Hoff, T. (2018, 10 febrero). La compleja infraestructura detrás de Netflix: ¿qué pasa cuando le das al «play»? Xataka. https://www.xataka.com/streaming/la-compleja-infraestructura-detras-de-netflix-que-pasa-cuando-le-das-al-play

López, D., & Maya, E. (2017). Arquitectura de software basada en microservicios para desarrollo de aplicaciones web.

Cárdenas Sánchez, B. C., & Olarte Rojas, C. A. (2022). Análisis de seguridad entre microservicios con Amazon Web Service. Revista Logos Ciencia & Tecnología, 14(2), 42-52.

De Paz Estrada, J. M. (2017). Diseño e implementación de una arquitectura escalable basada en micro servicios para un sistema de gestión de apr endizaje con caracteristicas de red social (Doctoral dissertation, Universidad de San Carlos de Guatemala).

Lu, S. X. (2023). Desarrollo y pruebas automáticas de microservicios sobre arquitectura Netflix para una aplicación web de integración de sistemas en el sector Portuario

Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.

Fowler, M., & Lewis, J. (2018). Microservices: Decomposing Applications for Deployability and Scalability. IEEE Software, 35(3), 82-86.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., & Mazzara, M. (2017). Microservices: Yesterday, Today, and Tomorrow. En European Conference on Service-Oriented and Cloud Computing (pp. 3-15). Springer.

Richardson, C. (2018). Microservices Patterns: With examples in Java. Manning Publications.

Fowler, M., & Lewis, J. (2018). Microservices: Decomposing Applications for Deployability and Scalability. IEEE Software, 35(3), 82-86.

Richardson, C. (2018). Microservices Patterns: With examples in Java. Manning Publications.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2011). Cloud computing and its relevance to grid computing. Future Generation Computer Systems, 27(6),599-616.

Debois, P., Humble, J., & Hunt, A. (2011). Twelve-factor app methodology.

Lewis, J. (2014). Microservices: a definition of this architectural style and some of its implications. ACM Queue, 12(3), 39-44

Microservicios 2.0: Spring Cloud Netflix vs Kubernetes & Istio - Paradigma (paradigmadigital.com)