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 1

ACT 1.1 DEFINIR LOS SIGUIENTES CONCEPTOS: MICROSERVICIOS

ALUMNO: MARCO ANTONIO ZÚÑIGA MORALES – A211121

DOCENTE: DR. LUIS GUTIÉRREZ ALFARO

TUXTLA GUTIÉRREZ, CHIAPAS
JUEVES, 17 DE AGOSTO DE 2023

ACT. 1.1 DEFINIR LOS SIGUIENTES CONEPTOS: MICROSERVICIOS

API:

Interfaz de Programación de Aplicaciones (API). Una API es un conjunto de reglas y protocolos que permiten que diferentes aplicaciones se comuniquen entre sí. Proporciona un conjunto de funciones y métodos predefinidos que los desarrolladores pueden utilizar para interactuar con un software, servicio o plataforma sin necesidad de conocer todos los detalles internos de su funcionamiento.

El término API es una abreviatura de Application Programming Interfaces, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

Una de las principales funciones de las API es poder facilitarles el trabajo a los desarrolladores y ahorrarles tiempo y dinero. Por ejemplo, si estás creando una aplicación que es una tienda online, no necesitarás crear desde cero un sistema de pagos u otro para verificar si hay stock disponible de un producto. Podrás utilizar la API de un servicio de pago ya existente, por ejemplo, PayPal, y pedirle a tu distribuidor una API que te permita saber el stock que ellos tienen.

Arquitectura:

Se refiere a la estructura organizativa y funcional de un sistema, aplicación o plataforma. Esto incluye la disposición de los componentes, la manera en que se comunican entre sí y cómo se cumplen los requisitos del sistema.

Arquitectura en informática es definida como el conjunto de estándares de tecnología de software y hardware que interactúan entre sí para formar un sistema o plataforma informática. En otras palabras, se refiere al proceso de diseño de sistemas informáticos para tecnologías compatibles.

AWS:

Amazon Web Services AWS es una plataforma de servicios en la nube ofrecida por Amazon. Proporciona una amplia gama de servicios, como almacenamiento, cómputo, bases de datos, análisis, aprendizaje automático y más, que permiten a las empresas y desarrolladores ejecutar aplicaciones y servicios en la nube.

AWS es la nube más adoptada y completa en el mundo, que ofrece más de 200 servicios integrales de centros de datos a nivel global. Millones de clientes, incluso las empresas emergentes que crecen más rápido, las compañías más

grandes y los organismos gubernamentales líderes, están usando AWS para reducir los costos, aumentar su agilidad e innovar de forma más rápida.

Ventaja

- **Elasticidad y escalabilidad:** AWS permite escalar recursos de manera eficiente según las necesidades cambiantes de tu aplicación o negocio. Puedes aumentar o disminuir la capacidad de cómputo, el almacenamiento y otros recursos de manera rápida y sencilla.

- **Pago por consumo:** AWS sigue un modelo de pago por uso, lo que significa que solo pagas por los recursos que realmente consumes. Esto elimina la necesidad de invertir en hardware costoso por adelantado y permite un mejor control de los costos.

- **Variedad de servicios:** AWS ofrece una amplia gama de servicios que abarcan cómputo, almacenamiento, bases de datos, análisis, machine learning, Internet de las cosas (IoT), seguridad y más. Esto permite a las empresas elegir los servicios que mejor se adapten a sus necesidades específicas.

- **Disponibilidad y confiabilidad:** AWS tiene una infraestructura global con múltiples centros de datos y regiones, lo que garantiza alta disponibilidad y redundancia. Además, ofrece servicios como Amazon S3 para almacenamiento altamente duradero y confiable.

- **Facilidad de uso:** AWS proporciona una interfaz de usuario intuitiva y una variedad de herramientas de administración para simplificar la configuración, el monitoreo y la administración de recursos en la nube.

- **Flexibilidad tecnológica:** AWS admite una amplia gama de sistemas operativos, lenguajes de programación, bases de datos y frameworks, lo que te permite elegir las tecnologías que mejor se adapten a tus aplicaciones.

- **Seguridad:** AWS ofrece una variedad de herramientas y servicios de seguridad para proteger tus datos y aplicaciones. Esto incluye cifrado, autenticación, control de acceso y cumplimiento de normativas.

- **Innovación continua:** AWS constantemente introduce nuevos servicios y mejoras para mantenerse a la vanguardia de la tecnología. Esto permite a las empresas aprovechar las últimas innovaciones sin tener que invertir en infraestructura nueva.

- **Globalización:** AWS tiene presencia en múltiples regiones y zonas de disponibilidad en todo el mundo, lo que facilita la expansión y la entrega de aplicaciones a nivel global.

- **Soporte técnico:** AWS ofrece diferentes niveles de soporte técnico según las necesidades de tu negocio, lo que incluye opciones de asistencia técnica las 24 horas del día, los 7 días de la semana.

Back End:

El backend se refiere a la parte de una aplicación o sistema que se encarga de las funciones y procesos detrás de escena. Incluye la lógica de negocios, el procesamiento de datos, la gestión de bases de datos y otros componentes que no son visibles para los usuarios finales.

Un backend es unos los sistemas corporativos que se utilizan para dirigir una web o empresa, tales como sistemas de gestión de pedidos, inventario y procesamiento de suministro. Este sistema recoge información de los usuarios u otros sistemas de tratamiento de datos en la compañía.

Las principales funciones del backend incluyen

- Optimizar las aplicaciones para que rindan mejor y de forma más rápida.
- Garantizar la seguridad de todo el entorno.
- Solucionar posibles errores que se vayan detectando.
- Gestionar e integrar las bases de datos necesarias.
- Trabajar mano a mano con el frontend developer.
- Realizar controles de calidad sobre el producto.
- Elaborar mejoras para las actualizaciones.
- Escribir códigos en diferentes lenguajes de programación.

Bifurcación:

Las bifurcaciones es la creación de un proyecto en una dirección distinta, cuando se aplica en el contexto de un lenguaje de programación o un sistema operativo, hace referencia a la creación de una copia de sí mismo por parte de un programa.

Se refiere a la creación de una versión independiente de un proyecto o repositorio. Esto permite que los desarrolladores trabajen en diferentes direcciones sin afectar la versión principal. Las bifurcaciones son comunes en plataformas de control de versiones como Git.

Escalabilidad:

La escalabilidad se refiere a la capacidad de un sistema o aplicación para manejar un aumento en la carga de trabajo o usuarios sin degradar su rendimiento. Puede

ser horizontal (agregando más máquinas) o vertical (mejorando los recursos de una máquina existente).

La escalabilidad en microservicios es crucial para garantizar un rendimiento adecuado y una respuesta rápida en una aplicación que está compuesta por varios servicios interconectados.

Algunas consideraciones clave sobre la escalabilidad en microservicios

- **Escalabilidad horizontal:** En lugar de simplemente mejorar la capacidad de una instancia de un microservicio, la escalabilidad en microservicios generalmente se logra agregando más instancias (escalabilidad horizontal). Esto permite distribuir la carga de trabajo entre múltiples instancias, lo que mejora el rendimiento y la capacidad de respuesta.

- **Desacoplamiento de servicios:** Dado que los microservicios son unidades independientes, puedes escalar solo aquellos que están experimentando una alta carga, en lugar de escalar toda la aplicación. Esto permite una utilización eficiente de recursos y una mejor capacidad para adaptarse a las variaciones en la demanda.

- **Orquestación y automatización:** Para lograr una escalabilidad eficiente en microservicios, es esencial contar con herramientas y plataformas de orquestación y automatización. Estas herramientas permiten implementar y escalar automáticamente nuevas instancias de microservicios según lo requiera la demanda.

- **Contenedores y orquestadores:** Las tecnologías de contenedores, como Docker, y los orquestadores, como Kubernetes, han sido fundamentales para lograr una escalabilidad efectiva en microservicios. Los contenedores proporcionan un entorno aislado y consistente para ejecutar microservicios, y los orquestadores facilitan la gestión, el despliegue y la escalabilidad de esos contenedores.

- **Monitoreo y ajuste:** Para lograr una escalabilidad óptima, es importante contar con herramientas de monitoreo y análisis que te permitan supervisar el rendimiento de los microservicios y ajustar la escalabilidad en función de la carga real y los patrones de uso.

Flexibilidad:

La flexibilidad se refiere a la capacidad de un sistema o aplicación para adaptarse y ajustarse a diferentes requisitos o cambios sin requerir cambios significativos en su estructura fundamental.

La computación flexible se refiere a la capacidad de maximizar el rendimiento activando y desactivando de manera selectiva los núcleos del procesador, lo que genera un posterior aumento o disminución de la frecuencia máxima de los núcleos activos restantes

Beneficios de tener flexibilidad en los sistemas de software

- Facilidad para el administrador
- Ahorro de costos
- Análisis optimizado
- Mayor seguridad
- Información en tiempo real
- Crecimiento e innovación

Framework:

Un framework es un conjunto de herramientas, bibliotecas y pautas que simplifican el desarrollo de software. Proporciona una estructura y funcionalidades comunes para ayudar a los desarrolladores a crear aplicaciones de manera más eficiente y consistente.

Un framework es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software.

Características

- **Reusabilidad:** Los frameworks permiten reutilizar componentes preconstruidos y soluciones probadas, lo que acelera el proceso de desarrollo y reduce la duplicación de esfuerzos.

- **Estructura:** Proporcionan una estructura organizativa para la aplicación, lo que ayuda a los desarrolladores a mantener un diseño coherente y escalable.

- **Abstracción:** Los frameworks abstraen aspectos técnicos complejos y proporcionan interfaces de programación de aplicaciones (API) simplificadas para tareas comunes.

- **Separación de preocupaciones:** Promueven la división de la lógica de negocios, la presentación y el acceso a datos, lo que facilita el mantenimiento y la colaboración entre equipos.

- **Convenciones:** Establecen prácticas y convenciones de codificación que ayudan a los desarrolladores a seguir estándares y buenas prácticas.

Ejemplos populares de frameworks

- **Ruby on Rails:** Framework de desarrollo web de alto nivel basado en Ruby.
- **Django:** Framework de desarrollo web de Python que enfatiza la eficiencia y la simplicidad.
- **Spring:** Framework de desarrollo de aplicaciones Java que se centra en la modularidad y la facilidad de integración.
- **React:** Framework de JavaScript para la construcción de interfaces de usuario interactivas.
- **Angular:** Framework de JavaScript mantenido por Google para el desarrollo de aplicaciones web.

Front End:

El frontend es la parte de una aplicación o sitio web con la que interactúan los usuarios finales. Incluye la interfaz de usuario, la presentación de datos y la interacción visual.

El frontend sirve para realizar la interfaz de un sitio web, desde su estructura hasta los estilos, como pueden ser la definición de los colores, texturas, tipografías, secciones, entre otros. Su uso es determinante para que el usuario tenga una buena experiencia dentro del sitio o aplicación.

Elementos del frontend

1. **Estructuras de navegación.** Este elemento se refiere al orden en que se organizan las diferentes páginas de un sitio web y a los componentes que se vinculan entre sí para realizar diferentes funciones dentro del sitio.

2. **Layout.** También nombrado diseño de página se refiere a todos los componentes de la página web, por ejemplo: ubicación del menú, botones, footer; todo lo necesario para que un sitio sea útil y fácil de navegar.

3. **Contenido web.** Todo aquello que brinde información relevante o interesante para los usuarios. Es importante destacar que el contenido no tiene que ser necesariamente texto, puede incluir sonido o materiales interactivos.

4. **Imágenes.** Todos los recursos visuales ayudan a aumentar el interés de los usuarios. Esto también puede incluir videos, animaciones, mapas, gráficas, infografías, GIFs, ilustraciones, diagramas, etc.

5. **Logotipo.** Para que un sitio web tenga mayor identidad es vital que contenga el logotipo que represente a la marca o empresa.

6. **Diseño gráfico.** Este elemento engloba todo lo relacionado con cómo se ve el sitio web y su apariencia: colores, formas, tipografías, tamaños, etc.

Aspectos clave a considerar al desarrollar el frontend en una arquitectura de microservicios:

- **Descomposición de la Interfaz de Usuario:** En lugar de tener un frontend monolítico que abarque toda la aplicación, se pueden crear interfaces de usuario separadas para cada microservicio. Esto significa que cada microservicio puede tener su propia interfaz de usuario dedicada.

- **Desacoplamiento:** Cada microservicio tiene su propia lógica de negocio y puede estar construido con diferentes tecnologías y lenguajes de programación. El frontend debe comunicarse con los microservicios a través de APIs bien definidas.

- **Carga Asimétrica:** Algunos microservicios pueden ser más intensivos en términos de tráfico y datos que otros. El frontend debe estar diseñado para manejar y balancear la carga de manera eficiente entre los microservicios.

- **Comunicación con los Microservicios:** La comunicación entre el frontend y los microservicios generalmente se realiza a través de solicitudes HTTP o API REST. Pueden utilizarse bibliotecas de cliente como Axios o Fetch para realizar estas solicitudes.

- **Enrutamiento y Navegación:** En una arquitectura de microservicios, la navegación y el enrutamiento entre diferentes partes de la aplicación pueden ser más complejos, ya que cada microservicio puede tener su propia ruta y lógica de navegación.

- **Gestión de Estado:** La gestión del estado en la interfaz de usuario puede ser un desafío, ya que el frontend puede tener que interactuar con varios microservicios para obtener y mostrar datos. Las bibliotecas de gestión del estado como Redux o MobX pueden ser útiles en este sentido.

- **Consistencia de Diseño:** Aunque cada microservicio puede tener su propio frontend, es importante mantener cierta coherencia en el diseño y la experiencia del usuario en toda la aplicación.

IaaS:

Infraestructura como servicio (IaaS), es un modelo de servicio en la nube que ofrece recursos de infraestructura bajo demanda, como computación, almacenamiento, redes y virtualización, a empresas y particulares a través de la nube.

La infraestructura como servicio (IaaS) consiste en alquilar servicios de infraestructura en la nube como servicios individuales de un proveedor de servicios en la nube, incluidos servidores, máquinas virtuales, recursos de redes y almacenamiento. La infraestructura como servicio (IaaS) ayuda a eliminar gran parte de la complejidad y los costes asociados a la construcción y al mantenimiento de la infraestructura física en un centro de datos on-premise.

Ventajas de la infraestructura como servicio

- **Ahorro de costes:** La infraestructura como servicio (IaaS) te ayuda a reducir tus gastos de capital iniciales. Los recursos se utilizan bajo demanda, por lo que solo tienes que pagar por los recursos de computación, almacenamiento y redes que consumas. Los costes de infraestructura como servicio (IaaS) son bastante predecibles, y se pueden incluir y presupuestar fácilmente.

- **Eficiencia mejorada:** Los recursos de IaaS suelen estar disponibles para las empresas cuando los necesitan. Gracias a esto, las organizaciones pueden reducir los retrasos en el aprovisionamiento cuando amplían la infraestructura y, de forma alternativa, evitar desperdiciar recursos aumentando la capacidad.

- **Más innovación:** Los equipos de TI no solo tienen más tiempo para invertir en el trabajo estratégico que les permite utilizar IaaS de forma rápida y asequible para probar nuevos productos e ideas. Puedes poner en marcha fácilmente la infraestructura informática necesaria sin tener que esperar días o semanas hasta que esté lista, lo que acelera los ciclos de vida de desarrollo y el tiempo de lanzamiento.

- **Fiabilidad:** Las plataformas de infraestructura como servicio no tienen un solo punto de fallo. La infraestructura en la nube ofrece redundancia y tolerancia a fallos integradas, ya que las cargas de trabajo se reparten entre varios servidores e instalaciones. Por lo general, el servicio seguirá estando disponible, aunque falle algún componente de los recursos de hardware.

- **Alta escalabilidad:** Una de las mayores ventajas de la infraestructura como servicio de computación en la nube es la capacidad de escalar recursos verticalmente rápidamente. Puedes adaptarse a los picos de demanda repentinos y casi al instante cuando los recursos ya no sean necesarios.

- **Latencia más baja:** La mayoría de los proveedores de servicios en la nube consiguen una mayor disponibilidad y resiliencia con la ayuda de una red global que abarca varias ubicaciones geográficas. Puedes minimizar la latencia y aumentar el rendimiento al colocar las aplicaciones y los servicios en las regiones y zonas más cercanas a tus usuarios finales.

Microservicios:

Los microservicios son un enfoque arquitectónico en el desarrollo de aplicaciones en el que una aplicación grande se divide en múltiples servicios pequeños e independientes. Cada microservicio se enfoca en una tarea específica y se puede desarrollar, implementar y escalar de manera individual. Esto permite una mayor flexibilidad y agilidad en el desarrollo y mantenimiento de aplicaciones complejas.

Los microservicios son módulos ligeros con acople suelto que pueden servir como componentes básicos de las aplicaciones complejas basadas en la nube. Aunque los microservicios individuales pueden operar de forma independiente, tienen un acople suelto en una interfaz unificada.

Características

- **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.

PaaS:

Plataforma como Servicio (PaaS), es un tipo de modelo de servicio de cloud computing que ofrece una plataforma en la nube flexible y escalable para desarrollar, desplegar, ejecutar y gestionar aplicaciones.

La PaaS proporciona a los desarrolladores todo lo que necesitan para desarrollar aplicaciones sin los quebraderos de cabeza que conlleva actualizar el sistema operativo y las herramientas de desarrollo y sin tener que mantener el hardware. En su lugar, los proveedores de servicios externos proporcionan la plataforma o la plataforma como servicio en la nube.

La PaaS ayuda a las empresas a no tener que molestarse en instalar y migrar hardware o software, o en desarrollar o alojar nuevas aplicaciones personalizadas.

Ventajas de las PaaS

- **Tiempo de lanzamiento más rápido:** No hay que esforzarse tanto. Los desarrolladores tienen acceso instantáneo a una plataforma completa de desarrollo de aplicaciones que no tienen que crear ni gestionar, por lo que pueden dedicar más tiempo a desarrollar y desplegar.

- **Pocos requisitos de mantenimiento:** Gracias a las pilas de aplicaciones internas conllevan muchos quebraderos de cabezas, sobre todo en cuanto a actualizaciones. Con la plataforma como servicio, el proveedor se encarga de mantener todo actualizado y no tienes que preocuparte por el mantenimiento.

- **Precios rentables:** Los recursos de PaaS están disponibles bajo demanda, por lo que solo pagarás por lo que utilices. Las PaaS también ofrecen acceso a herramientas y funciones de desarrollo avanzadas que podrían ser demasiado caras para comprarlas directamente.

- **Escalabilidad fácil:** Ya no tienes que preocuparte por la capacidad. Las PaaS te permiten reducir verticalmente en periodos de poco tráfico o escalar verticalmente para hacer frente a los aumentos inesperados en la demanda.

- **Acceso flexible:** Los equipos de desarrollo y DevOps pueden acceder a herramientas y servicios de PaaS compartidos desde cualquier lugar y en cualquier dispositivo a través de una conexión a Internet.

- **Seguridad compartida:** Con la plataforma como servicio, el proveedor se encarga de proteger la infraestructura. La mayoría de los principales proveedores de servicios de PaaS también ofrecen directrices y prácticas recomendadas para desarrollar en sus plataformas.

Servicio:

Un servicio es una funcionalidad específica proporcionada por un software o plataforma que se puede acceder y utilizar a través de una interfaz, como una API. Los servicios pueden variar desde servicios web hasta servicios en la nube, cada uno ofreciendo un conjunto específico de funciones.

En el contexto de la arquitectura de microservicios, un "servicio" se refiere a una unidad funcional autónoma y autocontenida que representa una parte específica de una aplicación. Los servicios son los componentes fundamentales de una arquitectura de microservicios y están diseñados para ser independientes y colaborar entre sí para construir una aplicación completa. Cada servicio se enfoca en una función o característica específica y se ejecuta como un proceso separado.

Características clave de un servicio en una arquitectura de microservicios

- **Autonomía:** Cada servicio es autónomo y tiene su propia lógica de negocio, datos y funcionalidad. Esto significa que puede ser desarrollado, desplegado y actualizado de manera independiente de otros servicios.

- **Comunicación a través de APIs:** Los servicios se comunican entre sí a través de interfaces de programación de aplicaciones (APIs). Esto permite que diferentes servicios se integren y colaboren sin necesidad de conocer los detalles internos de cada uno.

- **Escalabilidad Independiente:** Cada servicio puede escalarse de manera independiente según la carga y los requisitos. Esto permite ajustar los recursos para cada servicio según sus necesidades específicas.

- **Tecnologías y Lenguajes Diversos:** Los servicios pueden estar construidos con diferentes tecnologías y lenguajes de programación, siempre y cuando puedan comunicarse a través de las interfaces definidas.

- **Despliegue y Actualización Independientes:** Los servicios pueden ser desplegados y actualizados sin afectar otros servicios. Esto facilita la entrega continua y la implementación ágil de nuevas características.

- **Responsabilidad Única:** Cada servicio se enfoca en una única responsabilidad o función. Esto ayuda a mantener un acoplamiento bajo entre los diferentes componentes del sistema.

- **Resiliencia:** Si un servicio falla, no debería afectar la operación de otros servicios. La arquitectura de microservicios promueve la resiliencia y la tolerancia a fallos a través de la redundancia y la gestión adecuada de errores.

- **Diseño Basado en Dominio:** Los límites de los servicios suelen ser definidos por límites de dominio, lo que significa que cada servicio abarca una parte específica del problema que la aplicación está resolviendo.

Servidor:

Un servidor es una computadora o sistema que proporciona recursos, servicios o datos a otras computadoras, conocidas como "clientes", a través de una red. Los servidores pueden servir contenido web, almacenar y administrar bases de datos, proporcionar servicios de correo electrónico y mucho más.

Los servidores desempeñan un papel crucial en la infraestructura tecnológica al gestionar solicitudes, almacenar datos y facilitar la comunicación y el intercambio de información.

Tipos de servidores

- **Servidor Web:** Proporciona páginas web y contenido a través de la World Wide Web. Los servidores web manejan solicitudes HTTP y envían archivos HTML, imágenes, videos y otros recursos a los navegadores de los usuarios.

- **Servidor de Correo Electrónico:** Administra el envío, recepción y almacenamiento de correos electrónicos. Puede ser un servidor de correo entrante (POP3 o IMAP) o un servidor de correo saliente (SMTP).

- **Servidor de Archivos:** Almacena y gestiona archivos que pueden ser accedidos y compartidos por usuarios o dispositivos en una red. Ayuda a mantener la organización y seguridad de los datos.

- **Servidor de Bases de Datos:** Administra bases de datos y proporciona acceso a datos almacenados en ellas. Los servidores de bases de datos

pueden ser SQL (por ejemplo, MySQL, PostgreSQL) o NoSQL (por ejemplo, MongoDB, Cassandra).

- **Servidor de Aplicaciones:** Ejecuta aplicaciones y servicios que se pueden acceder desde clientes a través de una red. Los servidores de aplicaciones gestionan la lógica de negocio y la interacción con los usuarios.

- **Servidor DNS:** Traduce nombres de dominio legibles por humanos en direcciones IP numéricas utilizadas por las computadoras para localizar recursos en la red.

- **Servidor VPN:** Proporciona una red privada virtual que permite a los usuarios acceder de forma segura a recursos de red a través de conexiones encriptadas.

- **Servidor Proxy:** Actúa como intermediario entre los usuarios y otros servidores, mejorando la seguridad, el rendimiento y la administración del tráfico de red.

- **Servidor de Juegos:** Gestiona la lógica y las interacciones de los juegos en línea, permitiendo que los jugadores interactúen y jueguen juntos en tiempo real.

- **Servidor de Streaming:** Distribuye contenido multimedia, como video y audio en tiempo real, a través de Internet.

SOAP:

Protocolo Simple de Acceso a Objetos (SOAP), es un protocolo ligero para el intercambio de información en entornos descentralizados y distribuidos. Los mensajes SOAP son las transmisiones de información de remitentes a destinatarios. Los mensajes SOAP se pueden combinar para crear patrones de petición/respuesta.

SOAP es independiente del transporte, pero habitualmente se lleva a través de HTTP para ejecutarse con la infraestructura de Internet existente. SOAP habilita el enlace y la utilización de servicios Web descubiertos definiendo una vía de acceso de mensajes para direccionar mensajes. Se utiliza SOAP para consultar UDDI para servicios Web.

XML:

El lenguaje de marcado extensible (XML) permite definir y almacenar datos de forma compartible. XML admite el intercambio de información entre sistemas de computación, como sitios web, bases de datos y aplicaciones de terceros. Las reglas predefinidas facilitan la transmisión de datos como archivos XML a través de

cualquier red, ya que el destinatario puede usar esas reglas para leer los datos de forma precisa y eficiente.

El lenguaje de marcado extensible (XML) es un lenguaje de marcado que proporciona reglas para definir cualquier dato. A diferencia de otros lenguajes de programación, XML no puede realizar operaciones de computación por sí mismo.

Beneficios de usar XML

- Respaldo para las transacciones interempresariales
- Conservación de la integridad de los datos
- Mejora de la eficiencia de búsqueda
- Diseño de aplicaciones flexibles

Web Services:

Los servicios web son componentes de software que permiten que diferentes aplicaciones se comuniquen a través de la red utilizando estándares web como XML, SOAP, REST, entre otros. Proporcionan una forma estandarizada de intercambiar datos y funcionalidades entre sistemas heterogéneos.

Un web service o servicio web es un software con un formato basado en texto que funciona con Internet. Este sistema se encarga de permitir la transmisión de solicitudes y respuestas entre diferentes servidores o aplicaciones, sin importar las diferencias que existan entre los lenguajes de programación en el que fueron desarrolladas o la plataforma en la que se ejecutan.

Características de los servicios web

- Permite la interoperabilidad y el uso de multiplataformas.
- Su formato está basado en texto.
- Es una herramienta de fácil uso y acceso.
- Provee servicios integrados.
- Su alcance es global.
- Hace posible el intercambio de mensajes SOAP (Simple Object Access Protocol).
- Interfaz descrita en WSDL (Web Service Description Language).
- Se apoya en el formato HTTP (Protocolo de transferencia de hipertexto).

Web:

World Wide Web o la Web, es un sistema de información en línea que se accede a través de Internet. Consiste en páginas web interconectadas que contienen contenido multimedia, enlaces y documentos hipertexto. La web es uno de los pilares fundamentales de la era digital.

El sistema que nosotros conocemos hoy como "la Web" tiene varios componentes:

- El protocolo HTTP dirige las transferencias de datos entre el servidor y el cliente.
- Para acceder a un componente de la Web, el cliente proporciona un único identificador universal, llamado URL por sus siglas en inglés de Localizador Uniforme de Recursos (Uniform Resource Locator) o URI por sus siglas en inglés de Identificador Uniforme de Recursos (Uniform Resource Identifier) (formalmente llamado UDI por sus siglas en inglés de Identificador Universal de Documentos (Universal Document Identifier)).
- HTML por sus siglas en inglés de Lenguaje de Marcas de Hipertexto (Hypertext Markup Language) es el formato más común para publicar documentos web.

Enlazar, o conectar recursos a través de hyperlinks (hiperligas o hiperenlaces), es un concepto que define la Web, contribuyendo a su identidad como una colección de documentos conectados

Balanceador:

Un balanceador de carga es un dispositivo o software que se utiliza para distribuir de manera eficiente el tráfico entrante entre varios servidores o recursos. El objetivo principal es evitar la sobrecarga en un solo servidor y mejorar la disponibilidad y el rendimiento de una aplicación o sitio web.

Los balanceadores de carga pueden ser tanto hardware como software y se utilizan para garantizar que los usuarios tengan una experiencia fluida y rápida al acceder a un sitio web o servicio.

REST:

REST es un estilo arquitectónico que se utiliza en el diseño de servicios web y aplicaciones distribuidas. Se basa en una serie de principios que incluyen la separación de recursos en la web y la comunicación a través de solicitudes HTTP estándar como GET, POST, PUT y DELETE. REST enfatiza la simplicidad, la escalabilidad y la interoperabilidad entre sistemas heterogéneos.

REST no es un protocolo ni un estándar, sino más bien un conjunto de límites de arquitectura. Los desarrolladores de las API pueden implementarlo de distintas maneras.

Cuando el cliente envía una solicitud a través de una API de RESTful, esta transfiere una representación del estado del recurso requerido a quien lo haya solicitado o al extremo. La información se entrega por medio de HTTP en uno de estos formatos: JSON (JavaScript Object Notation), HTML, XLT, Python, PHP o texto sin formato.

REFERENCIA BIBLIOGRÁFICA

Bifurcaciones. (2017, March 27). INFORMATICA. https://gratis68584.wordpress.com/bifurcaciones/

Coppola, M. (2023, April 12). Frontend y backend. *https://blog.hubspot.es/website/frontend-y-backend*.

Descripción de la computación flexible - Guía de administración de los servidores Oracle® serie X5. (2015, August 12). https://docs.oracle.com/cd/E50691_01/html/E58595/goibb.html#:~:text=La%20computaci%C3%B3n%20flexible%20se%20refiere,de%20los%20n%C3%BAcleos%20activos%20restantes.

Edix, R. (2022a, July 26). *Backend developer: que es, funciones y cómo serlo*. Edix España. https://www.edix.com/es/instituto/backend-developer/#:~:text=%C2%BFQu%C3%A9%20es%20un%20backend%20developer,todo%20para%20su%20correcto%20funcionamiento.

Edix, R. (2022b, July 26). *Framework: qué es, para qué sirve y algunos ejemplos*. Edix España. https://www.edix.com/es/instituto/framework/#:~:text=Un%20framework%20es%20un%20esquema,organizaci%C3%B3n%20y%20desarrollo%20de%20software.

Fernández, Y. (2019). API: qué es y para qué sirve. *Xataka*. https://www.xataka.com/basics/api-que-sirve

García, I. J. B. (2021, March 30). *Backend y Frontend, ¿Qué es y cómo funcionan en la programación?* https://www.servnet.mx/blog/backend-y-frontend-partes-fundamentales-de-la-programacion-de-una-aplicacion-web

¿Qué es AWS? (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is-aws/

¿Qué es IaaS (infraestructura como servicio)? | Google Cloud. (n.d.). Google Cloud. https://cloud.google.com/learn/what-is-iaas?hl=es#:~:text=La%20infraestructura%20como%20servicio%20(IaaS,a%20trav%C3%A9s%20de%20la%20nube.

¿Qué es la arquitectura informática? (n.d.). SAINT LEO. https://worldcampus.saintleo.edu/noticias/que-es-la-arquitectura-informatica

¿Qué es una PaaS? | Google Cloud. (n.d.). Google Cloud. https://cloud.google.com/learn/what-is-paas?hl=es#:~:text=servicio%20(PaaS)%3F-,%C2%BFQu%C3%A9%20es%20una%20plataforma%20como%20servicio%20(PaaS)%3F,desplegar%2C%20ejecutar%20y%20gestionar%20aplicaciones.

¿Qué es XML? - Explicación del lenguaje de marcado extensible (XML) - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/xml/#:~:text=El%20lenguaje%20de%20marcado%20extensible%20(XML)%20es%20un%20lenguaje%20de,de%20computaci%C3%B3n%20por%20s%C3%AD%20mismo.

Qué son los microservicios y la arquitectura de microservicios? -. . . (n.d.). Intel. https://www.intel.la/content/www/xl/es/cloud-

computing/microservices.html#:~:text=Los%20microservicios%20son%20m%C3%B3dulos%20ligeros%20que%20pueden%20servir%20como%20componentes,suelto%20en%20una%20interfaz%20unificada.

Ramirez, N. (2022). BENEFICIOS DE TENER FLEXIBILIDAD EN LOS SISTEMAS DE SOFTWARE DE RRHH. *VenturesSoft*. https://venturessoft.com/beneficios-de-tener-flexibilidad-en-los-sistemas-de-software-de-rrhh/

REST API. (n.d.). https://www.redhat.com/es/topics/api/what-is-a-rest-api

Servidor en Informática - Concepto, características y ejemplos. (n.d.). Concepto. https://concepto.de/servidor/

Urrutia, D. (2021). Qué es Backend - Definición, significado y ejemplos. *Arimetrics*. https://www.arimetrics.com/glosario-digital/backend#:~:text=Definici%C3%B3n%3A,de%20datos%20en%20la%20compa%C3%B1%C3%ADa.

What is Software Load Balancing? | *VMware Glossary*. (2022, September 20). VMware. https://www.vmware.com/es/topics/glossary/content/software-load-balancing.html#:~:text=El%20balanceo%20de%20carga%20definido%20por%20software%20es%20la%20forma,el%20contenido%20de%20la%20solicitud).

World Wide Web - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN. (n.d.). https://developer.mozilla.org/es/docs/Glossary/World_Wide_Web