FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN CAMPUS 1

ING. EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE

6 “M”

Materia: TALLER DE DESARROLLO 4

Act. 1.1 Definir los siguientes conceptos: Microservicios

ALUMNO: Luis Antonio Castro Gutiérrez

Matricula: A211125

DOCENTE: DR. LUIS GUTIÉRREZ ALFARO

TUXTLA GUTIÉRREZ, CHIAPAS

SÁBADO, 18 DE AGOSTO DE 2023

API:

Un API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y protocolos que permite a diferentes aplicaciones y sistemas comunicarse y compartir datos de manera organizada y segura. Actúa como un intermediario que permite que una aplicación utilice funciones y servicios de otra sin necesidad de conocer los detalles internos de su funcionamiento. Los API definen cómo las solicitudes y respuestas deben estructurarse, lo que facilita la interacción entre programas. Esto es esencial en el desarrollo de software moderno, ya que permite la integración eficiente de componentes y servicios, fomentando la colaboración y la construcción rápida de aplicaciones más poderosas y versátiles.

Arquitectura:

La arquitectura se refiere al diseño y estructura de un sistema o aplicación, organizando la interacción y comunicación entre los componentes del software para lograr un objetivo específico. En el caso de las API, la arquitectura define cómo se diseñan las interfaces de programación, incluyendo solicitudes, respuestas, autenticación, autorización y manejo de errores. Una arquitectura de API sólida facilita la integración entre sistemas y el desarrollo escalable de aplicaciones. En el desarrollo de software en general, la arquitectura se enfoca en la organización y diseño del código, componentes y capas para crear un sistema funcional y sostenible. Diferentes enfoques arquitectónicos, como capas, SOA y microservicios, ofrecen diversas ventajas y desafíos.

AWS:

Amazon Web Services (AWS) es una plataforma de servicios de nube proporcionada por Amazon que ofrece una amplia gama de servicios de infraestructura, almacenamiento, cómputo, bases de datos, análisis, inteligencia artificial y más. AWS permite a las organizaciones y desarrolladores utilizar recursos informáticos bajo demanda sin tener que invertir en hardware físico y administrar infraestructuras complejas.

Ventajas de AWS:

- Elasticidad y escalabilidad
- Amplia gama de servicios
- Pago según uso
- Disponibilidad y durabilidad
- Rápida implementación
- Seguridad
- Facilidad de integración
- Escalabilidad global
- Flexibilidad tecnológica
- Innovación continua

Desventajas de AWS:

- Costos potenciales

- Complejidad
- Dependencia de la nube
- Latencia
- Seguridad compartida
- Riesgo de interrupciones
- Regulaciones y cumplimiento
- Capacitación y habilidades
- Bloqueo de proveedor
- Privacidad de datos

Backend:

El backend, o lado del servidor, gestiona la lógica y datos no visibles para usuarios. Procesa solicitudes, interactúa con bases de datos, hace cálculos, y proporciona información al frontend. Engloba:

- Gestión de datos: Almacena y manipula información en bases de datos.
- Lógica del negocio: Implementa reglas y procesos de la aplicación.
- Procesamiento: Realiza cálculos y tareas computacionales.
- Seguridad: Controla accesos y autenticación para proteger datos.
- Integración: Conecta con sistemas externos a través de API.

Bifurcación:

También conocida como "branching", se refiere a una técnica que permite que un programa tome diferentes caminos de ejecución en función de una condición o decisión. En este contexto, se utilizan estructuras de control como las declaraciones condicionales (if, else if, else) y los bucles para determinar qué bloques de código se ejecutan en diferentes situaciones. La bifurcación es esencial para crear programas flexibles y reactivos, ya que permite adaptarse a diferentes escenarios. Los resultados de la bifurcación pueden ser acciones diferentes, cálculos alternativos o rutas de flujo variadas en un programa. Esta capacidad de tomar decisiones y ajustar el comportamiento del programa en función de las condiciones es fundamental para la creación de aplicaciones más sofisticadas y eficientes.

Escalabilidad:

Hace referencia a la capacidad de un sistema o aplicación para manejar un aumento en la carga de trabajo o la demanda sin comprometer su rendimiento, disponibilidad o calidad. Implica diseñar y construir sistemas de manera que puedan adaptarse y crecer de manera eficiente en respuesta a un aumento en la cantidad de usuarios, datos o transacciones.

En programación, la escalabilidad se logra a través de técnicas como la optimización de algoritmos, el uso eficiente de recursos (como memoria y procesamiento), la distribución de cargas en múltiples servidores (escalabilidad horizontal) o la mejora de componentes individuales (escalabilidad vertical). La falta de escalabilidad puede resultar en tiempos de respuesta lentos, interrupciones del servicio y una mala experiencia del usuario. Por lo tanto, es esencial considerar la escalabilidad desde las primeras etapas de diseño y

desarrollo de un sistema para asegurar su capacidad de crecimiento y adaptación a medida que las necesidades cambian con el tiempo.

Flexibilidad:

la flexibilidad se refiere a la capacidad de un sistema, componente o código para adaptarse y responder eficientemente a cambios en los requisitos, condiciones o entornos sin requerir modificaciones extensas o causar efectos no deseados en otras partes del sistema. Una estructura flexible está diseñada de manera modular, con componentes independientes y cohesivos que pueden reconfigurarse o extenderse sin afectar negativamente el funcionamiento global. La flexibilidad se logra mediante principios como la separación de preocupaciones, la abstracción y la encapsulación. Además, el uso de patrones de diseño y prácticas como la inyección de dependencias y la programación orientada a interfaces ayuda a crear sistemas flexibles, permitiendo una adaptación rápida y eficaz a los cambios en los requerimientos o en el entorno tecnológico.

Framework:

Los frameworks también ofrecen un conjunto de reglas y convenciones que ayudan a mantener un código organizado y coherente en todo el proyecto. Al definir una estructura preestablecida, los desarrolladores pueden trabajar en equipo de manera más eficiente, ya que todos siguen las mismas directrices. Además, los frameworks a menudo facilitan la implementación de patrones de diseño comunes, lo que mejora la calidad y la mantenibilidad del código.Otra ventaja clave es la comunidad activa que rodea a muchos frameworks, lo que brinda acceso a documentación detallada, tutoriales y soporte en línea. Esto acelera el aprendizaje y resolución de problemas, especialmente para aquellos nuevos en el framework.

En resumen, los frameworks son herramientas esenciales en el desarrollo de software moderno, ya que ahorran tiempo, mejoran la estructura del código y permiten a los desarrolladores centrarse en la creación de funcionalidades específicas en lugar de enfrentar desafíos recurrentes. Su adopción puede impulsar la productividad, la calidad y la innovación en el desarrollo de aplicaciones.

Front-end

El frontend, llamado también el lado del cliente, es la parte visible de una aplicación que los usuarios finales interactúan directamente. Su función es presentar la información de forma visual y comprensible, permitiendo la interacción mediante interfaces gráficas.

Este enfoque se centra en la experiencia del usuario, incluyendo el diseño de la interfaz, disposición de elementos, navegación e interacción. Emplea lenguajes como HTML para estructurar el contenido, CSS para estilo visual, y JavaScript para funcionalidades dinámicas.

El frontend cuenta con algunas ventajas:

- Diseña la interfaz de usuario con elementos visuales para interacción.

- Mejora la experiencia del usuario para usabilidad y accesibilidad.
- Define apariencia visual mediante hojas de estilo.
- Agrega dinamismo y respuesta usando JavaScript.
- Facilita la navegación y estructura de la aplicación para una experiencia intuitiva.

IaaS:

IaaS (Infrastructure as a Service) es un modelo en la nube que suministra infraestructura informática virtualizada a través de la web. En lugar de comprar y gestionar hardware físico, las organizaciones arriendan recursos como servidores virtuales, almacenamiento y redes desde un proveedor de la nube.

Los usuarios tienen control sobre la configuración, administración y escalabilidad de estos recursos virtuales, lo que les permite implementar sistemas operativos y aplicaciones, adaptando la configuración a sus necesidades. Ejemplos de proveedores incluyen AWS, Azure, GCP y IBM Cloud.

IaaS ofrece ventajas como flexibilidad para ajustar recursos según demanda, reducción de costos iniciales y agilidad en la implementación. Es idóneo para empresas que buscan soluciones ágiles y rentables para sus necesidades de infraestructura, sin requerir inversión en hardware y mantenimiento físico. La adopción de IaaS posibilita concentrarse en funciones de alto valor en lugar de gestionar la infraestructura, propiciando la eficiencia y permitiendo un crecimiento escalable de manera sencilla.

Microservicios:

Los microservicios son un enfoque arquitectónico en el desarrollo de software donde una aplicación se divide en componentes pequeños e independientes, conocidos como microservicios. Cada microservicio se centra en una función específica y opera de manera autónoma. Estos microservicios se comunican a través de interfaces definidas, como APIs, para construir una aplicación completa. Los microservicios permiten una mayor flexibilidad, escalabilidad y mantenibilidad en comparación con arquitecturas monolíticas. Cada microservicio puede ser desarrollado, implementado y escalado individualmente, lo que facilita cambios ágiles y despliegues continuos. Aunque ofrecen ventajas en términos de modularidad y resiliencia, también requieren una gestión cuidadosa de la comunicación entre componentes y la coordinación en el ecosistema de microservicios.

PaaS:

PaaS (Platform as a Service) es un modelo de servicios en la nube que proporciona a los desarrolladores un entorno completo para construir, implementar y administrar aplicaciones sin preocuparse por la infraestructura subyacente. En lugar de lidiar con la configuración y el mantenimiento de servidores, redes y sistemas operativos, los usuarios de PaaS se centran en la creación de aplicaciones y servicios.

En un entorno PaaS, se ofrece una plataforma de desarrollo que incluye herramientas, bibliotecas y servicios preconfigurados para facilitar el proceso de creación de software.

Los desarrolladores pueden aprovechar estas herramientas para escribir código, implementar aplicaciones, administrar bases de datos y automatizar tareas, todo dentro del mismo entorno.

Servicio:

los servicios son componentes o módulos que proporcionan funcionalidades específicas y son accesibles a través de la red, generalmente utilizando protocolos como HTTP. Estos servicios permiten la interacción y comunicación entre diferentes aplicaciones, sistemas o dispositivos de manera eficiente y estandarizada. Los servicios web, por ejemplo, utilizan estándares como XML o JSON para estructurar y transmitir datos, y pueden ofrecer diversas operaciones como consultar información, realizar cálculos o enviar datos.

Los servicios en programación web pueden abarcar una amplia gama de funciones, desde autenticación y autorización hasta procesamiento de pagos, almacenamiento de archivos, envío de correos electrónicos, y más. Al separar estas funcionalidades en servicios independientes, se promueve la modularidad y la reutilización de código, lo que resulta en un desarrollo más eficiente y una arquitectura más escalable.

Servidor:

Un servidor es una computadora o sistema informático que proporciona recursos, servicios o datos a otras computadoras, dispositivos o usuarios en una red. Actúa como un punto centralizado para almacenar, procesar y entregar información solicitada por los clientes, que pueden ser otras computadoras, aplicaciones o usuarios finales.

Los servidores pueden desempeñar diferentes roles y funciones según el contexto:

- Servidor web
- Servidor de correo electrónico
- Servidor de bases de datos
- Servidor de archivos
- Servidor de aplicaciones
- Servidor de impresión

SOAP:

SOAP (Simple Object Access Protocol) es un protocolo de comunicación utilizado en servicios web para intercambiar información estructurada en formato XML a través de redes, generalmente utilizando el protocolo HTTP. Fue diseñado para permitir que aplicaciones en diferentes plataformas y lenguajes se comuniquen de manera interoperable.

Características de SOAP:

Estructura XML: SOAP utiliza XML como formato de intercambio de datos, lo que facilita la representación de información estructurada y legible tanto para humanos como para máquinas.

Independencia de plataforma y lenguaje: SOAP permite que aplicaciones escritas en diferentes lenguajes de programación y que se ejecutan en diferentes plataformas puedan comunicarse entre sí, ya que se basa en estándares y protocolos abiertos.

Protocolo extensible: SOAP permite la definición de reglas personalizadas y extensiones para adaptarse a requerimientos específicos de las aplicaciones, lo que aumenta su flexibilidad y adaptabilidad.

XML:

XML (Extensible Markup Language) es un lenguaje de marcas utilizado para estructurar y almacenar datos de manera legible tanto para humanos como para máquinas. Se basa en etiquetas (marcas) que definen la estructura jerárquica de los datos y sus metadatos, lo que permite la representación organizada y la transferencia de información entre sistemas y aplicaciones.

Las características de XML son:

- Legibilidad: XML utiliza etiquetas con nombres descriptivos para representar elementos y atributos, lo que facilita la comprensión y el análisis de los datos.
- Extensibilidad: XML permite definir etiquetas personalizadas y estructuras de datos específicas para adaptarse a diferentes necesidades y dominios.
- Interoperabilidad: Al ser independiente de la plataforma y del lenguaje, XML facilita el intercambio de datos entre aplicaciones heterogéneas.
- Jerarquía: Los datos se organizan en una estructura de árbol jerárquico compuesta por elementos, atributos y contenido.
- Principio del formulario

Web services:

Los servicios web son una tecnología que permite a aplicaciones y sistemas comunicarse y compartir datos a través de Internet o redes. Funcionan como bloques de construcción que ofrecen funcionalidades específicas a otras aplicaciones, independientemente de su plataforma o lenguaje de programación. Estos servicios utilizan estándares como HTTP y XML para transmitir información de manera interoperable. Dos tipos comunes de servicios web son SOAP (Simple Object Access Protocol) y REST (Representational State Transfer). SOAP utiliza XML para estructurar datos y se centra en la seguridad y la integridad, mientras que REST utiliza HTTP para acceder a recursos a través de URLs y se destaca por su simplicidad y escalabilidad. Los servicios web juegan un papel clave en la integración de sistemas y la construcción de aplicaciones distribuidas.

Web:

La web, o World Wide Web, es un sistema global de información interconectado en Internet, creado por Tim Berners-Lee en los años 80. Se compone de páginas web con texto, imágenes, videos y enlaces, permitiendo a los usuarios explorar contenido no linealmente a través de hipervínculos. Navegadores como Chrome y Firefox permiten acceder a la web.

Con el tiempo, la web ha evolucionado, introduciendo tecnologías como HTML para estructurar contenido, CSS para diseño visual y JavaScript para interactividad. Ha dado lugar a aplicaciones web avanzadas, servicios en línea y comercio electrónico, transformando la manera en que las personas acceden a la información y se conectan globalmente.

Balanceador:

Un balanceador de carga es una herramienta esencial que equilibra eficientemente la carga del tráfico entrante entre varios servidores o recursos. Su objetivo principal es prevenir la saturación de un único servidor, mejorando la disponibilidad y el rendimiento de aplicaciones o sitios web.

Estos balanceadores pueden ser hardware o software, y su función clave es asegurar que los usuarios experimenten una navegación ágil y veloz al acceder a un sitio web o servicio. Al distribuir el tráfico de manera equitativa, evitan la congestión y mejoran la capacidad de respuesta, lo que garantiza una experiencia de usuario positiva y estable. En resumen, los balanceadores de carga son una solución esencial para optimizar la utilización de recursos y mantener la eficacia en entornos digitales de alta demanda.

REST

REST (Representational State Transfer) es un estilo arquitectónico utilizado en el diseño y desarrollo de servicios web y aplicaciones que se comunican a través de la web. A diferencia de SOAP, que es un protocolo, REST es un conjunto de principios y restricciones que guían la creación de sistemas web eficientes, escalables y mantenibles.

Las principales características de REST incluyen:

Recursos y URI: En REST, los componentes de un sistema se modelan como recursos (entidades) que se identifican mediante URIs (Uniform Resource Identifiers), como URLs. Cada recurso se accede mediante su URI única.

Métodos HTTP: REST utiliza los métodos estándar de HTTP, como GET (para obtener información), POST (para crear nuevos recursos), PUT (para actualizar recursos) y DELETE (para eliminar recursos), para realizar acciones en los recursos.

Estado representacional: Las respuestas de los servicios REST contienen representaciones de los recursos en formatos como XML o JSON. Cada respuesta debe incluir suficiente información para entender y manipular el recurso.

Sin estado: Cada solicitud a un servicio REST debe contener toda la información necesaria para comprender y procesar la solicitud, lo que significa que el servidor no mantiene información sobre el estado de las solicitudes anteriores.

Referencias:

Aguirre, S. (2022). Crea tu API (Vol. 202). RedUSERS.

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.

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

Pérez Ibarra, S. G., Quispe, J. R., Mullicundo, F. F., & Lamas, D. A. (2021). Herramientas y tecnologías para el desarrollo web desde el FrontEnd al BackEnd. In XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja).

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

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

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

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

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