|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
| Nombre del Maestro(a): Hugo Eden Moroyoqui Alvarado | | | Calificación:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
|  |  |  | |  |
| Nombre de Alumno(a): Victor Galvan |  | Fecha: 24 de enero de 2022 | |  |

**Actividad #3: Resumen**

Desarrolla un resumen de las partes más importantes de cada presentación de estilos de arquitecturas.

Enumera cada resumen de modo que al final tengamos un total de 6 resúmenes de cada tema expuesto.

**Equipo #1 (Isaac, Axel y Sebastián) - Cliente-Servidor**

La arquitectura cliente-servidor es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.

Características

* El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas.
* Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.
* Cada plataforma puede ser escalable independientemente.
* La interrelación entre el hardware y el software está basada en una infraestructura poderosa.
* Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina.

Herramientas

* Cliente: Programa ejecutable que participa activamente en el establecimiento de las conexiones.
* Servidor: Es un programa que ofrece un servicio que se puede obtener en una red. Acepta la petición desde la red, realiza el servicio y devuelve el resultado al solicitante.
* Los servidores: pueden ejecutar tareas sencillas o complejas.
* Los servidores por lo general tienen dos partes:
  + Programa o proceso que es responsable de aceptar nuevas peticiones: Maestro o Padre.
  + Programas o procesos que deben manejar las peticiones individuales: Esclavos o Hijos.

Ventajas

* Centralización del control
* Escalabilidad
* Fácil mantenimiento.
* Existen tecnologías suficientemente desarrolladas.
* Se dificulta el rastreo y/o hackeo de los usuarios.

Desventajas

* La gran cantidad de información que viaja a través de los clientes sobre congestiona el tráfico, le que significa bajo rendimiento.
* Debido a su bajo rendimiento, esta estructura tiene un bajo espectro de aplicación, limitándose a la construcción de sistemas no críticos.
* Tanto la instalación como el mantenimiento son mayores debido al perfil altamente técnico del lado del servidor.

**Equipo #2 (Adrián y Erick) - Micro servicios**

Un estilo de arquitectura y para programar software donde las aplicaciones se dividen en elementos pequeños elementos independientes enfocados a la misma tarea y se valora el nivel de detalle, sencillez y capacidad para compartir procesos similares.

Características

* En software: puede ser descompuesto.
* En su organización: Tienen en cuenta capacidades, necesidades y preferencias del cliente.
* En su arquitectura: Cada módulo independiente cuenta con su propia base de datos.
* En su sistema de aviso y actuación: Envía advertencias si hay algún fallo.

Ventajas

* Escalabilidad
* Implementación sencilla
* Código reutilizable
* Agilidad de cambios
* Aplicación independiente
* Menor riesgo

Desventajas

* Memoria
* Tiempo para fragmentar
* Dificultad para gestionar
* Solucionar problemas de latencia
* Testeos complicados

**Equipo #3 (Fernando Osuna y Fernando Jr.) - N capas**

La programación por capas es una arquitectura cliente-servidor en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades.

El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada.

Características

* Capa de presentación
  + Presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso. También es conocida como interfaz gráfica.
* Capa de negocios
  + Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse.
* Capa de datos
  + Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Ventajas

* Portabilidad entre la nube y entornos locales, y entre plataformas en la nube.
* Menor curva de aprendizaje para la mayoría de los desarrolladores
* Evolución natural desde el modelo de aplicaciones tradicional.
* Abiertas a entornos heterogéneos (Windows o Linux)

Desventajas

* A veces no se logra la contención del cambio y se requiere una cascada de cambios en varias capas.
* Trabajo innecesario por parte de capas más internas o redundante entre varias capas.
* Dificultad de diseñar correctamente la granularidad de las capas.
* Pérdida de eficiencia.

**Equipo #4 (Carlos Olea y Carlos Porras) - Dirigida a eventos**

La arquitectura dirigida por eventos es un tipo de arquitectura software que se basa en la producción, detención y procesamiento de eventos y, si se considera oportuno, la reaccionante ellos. En la mayoría de los casos, el dispositivo que genera un evento no es el mismo que el que lo recibe y procesa.

Características

* Generador del evento:
  + Es la fuente que arroja el evento, cualquier tipo de dispositivo o aplicación podría ser el generador.
* Canal del evento:
  + Es el medio de unión entre el generador y el procesador, su objetivo es transportar las notificaciones.
* Procesador del evento:
  + Es el servidor que recibe los eventos, los procesa, los compara con reglas preestablecidas y realiza las acciones.
* Actividad de respuesta:
  + Es la acción que se hace si se cumple alguna de las reglas, puede ser cualquier tipo de acción.

Ventajas

* Desacoplo de servidor y clientes.
* Desacoplo entre dispositivos.
* Altamente escalable.
* Arquitectura dirigida por eventos es asíncrona sin bloqueo.
* Los servicios no necesitan saber o depender de otros servicios.

Desventajas

* El servidor no podría saber nunca si un servicio se ha caído.
* No se puede comprobar si un evento ha sido atendido.

**Equipo #5 (Karina y David) – Microkernel**

Conocido como arquitectura de Plug-in, permite crear aplicaciones extensibles, mediante la cual es posible agregar nuevas funcionalidades mediante la adición de pequeños plugins que extienden la funcionalidad inicial del sistema.

Herramientas

* Eclipse
* Netbeans
* Visual Studio Code
* sistemas de ofimática como:
  + Word
  + Powerpoint
  + Excel

Ventajas

* Testabilidad: Debido a que los Plugins y el sistema Core son desarrollados de forma por separado, es posible probarlos de forma aislada.
* Performance: En cierta forma, muchas de las aplicaciones basadas en Microkernel trabajan de forma Monolítica una vez que el Plug-in es instalado, lo que hace que todo el procesamiento se haga en una sola unidad de software.
* Despliegue: Debido a la naturaleza de Plugins es posible instalar fácilmente todas las características adicionales que sea necesarias, incluso, pueden ser agregar en tiempo de ejecución, lo que en muchos casos ni siquiera requiere de un reinicio del sistema.
* Reutilización: Debido a que los Plugins puede ser instalados en cualquier instancia del sistema Core, es posible reutilizar los módulos en varias instancias, incluso, es posible comercializarlas de forma independiente. Solo como ejemplo, existen empresas que se dedican exclusivamente a desarrollar Plugins para venderlos, como es el caso de los Plugins de Wordpress.

Desventajas

* Los sistemas que utilizan una arquitectura de Microkernel no son fáciles de desarrollar.
* El sistema Core y los plugins son desarrollados por lo general por equipos separados.

**Equipo #6 (Victor y Tania) – Serverless**

Una arquitectura sin servidor es una manera de crear y ejecutar aplicaciones y servicios sin tener que administrar infraestructura.

Servicios terceros

* Mismo proveedor
* Servicios de proveedor distintos

Características

* El comportamiento se ensambla a partir de servicios, el control del flujo se hace del lado del cliente y el contenido se genera de forma dinámica en el cliente.
* Las aplicaciones serverless el ciclo de vida de los programas es mucho más corto, tendiendo a ser un solo ciclo petición/respuesta en HTTP.

Herramientas

* Docker-Lambda: Es un ecosistema de tipo sandbox (caja de arena) que replica todas las configuraciones y funcionalidades de la función Lambda, la cual es ofrecida por Amazon Web Services.
* Protego: Es una aplicación web que prueba los distintos aspectos de seguridad que se deben tener en cuenta durante todo el ciclo de desarrollo de una aplicación.
* Snyk: Mediante esta herramienta es posible automatizar tanto los procesos de mantenimiento como los relacionados con la seguridad de la aplicación en sí.

Ventajas

* Elimina la administración de infraestructura TI
* Resultan en una reducción de costos
* Permite enfocar el trabajo en lo que importa
* Facilita la adopción de ideas como soluciones reales

Desventajas

* Complejidad
* Carga y latencia
* Hacer test
* Configurar mucho tiempo de ejecución
* Controlar la arquitectura correctamente