**Introducción a la arquitectura de software**

¿Qué es arquitectura de software?

-No hay una definición formal entendida por toda la comunidad

**3 definiciones posibles:**

1.La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar acerca del sistema.

2.Es el conjunto de decisiones de diseño importantes para organizar el software y promover los atributos de calidad deseados.

3.La arquitectura es acerca de las cosas importantes sean las que sean(diseño a mas alto nivel y depende de quien la vea)}

Conceptos claves: Estructuras, Razonar

Estructrutas: Como se va a organizar cada una de las partes del sistema y como se van a conectar cada una de estas partes, para eso se usan los patrones de arquitectura.

Razonar: La idea es que sirva para que el equipo de desarrollo pueda tener discusiones del diseño del sistema al mas alto nivel

Decisiones de diseño: Se tiene que definir para la arquitectura cuales son las decisiones de diseño que van a guiar el trabajo del equipo. Deben ser tomadas y documentadas para ser comunicadas apropiadamente al equipo

Atributos de calidad: Permiten definir que es lo que va mas alla de lo funcional que va garantizar que el sistema funciones bien.

-Facilidad de mantenimiento

-Desplegabilidad

-Elasticidad

-Escalabilidad

2) Algunas actividades que hace un arquitecto de software

1. Define y balancea los atributos de calidad, entre mas atributos de calidad tenga, se debe tener unos y sacrificar otros si es necesario.

2. Define el problema desde un punto de vista de ingenieria. Toma todo el contexto plantea el problema y lo diseña a un alto nivel para resolverlo

3. Establece los principios de diseño que guían las decisiones de tecnología de una empresa. Debe ser capaz de definir linieamnientos que el equipo pueda seguir y poderlos guiar para que tomen decisiones

4. Garantizar el cumplimiento de la arquitectura, aunque define la arquitectura, debe involucrarse para que a medida que se implemente el diseño el este ahí garantizando que se cumpla.

5. Mantenerse al día con las ultimas tendencias tecnologicas. Saber que desarrollos se estan dando en la actualidad y sobre todo en la industria del negocio en la que esta ese sistema.

6. Conocer así sea de forma superficial distintas tecnologias como en front y back y movil, las entiende y puede tomar decisioens apartir de ese conocimiento.

7. Analiza el entorno y propone mejores para que la arquitectura de un sistema se mantenga relevante. Propone mejoras a la arquitectura.

8. Adiministrar la deuda tecnica: Es la diferencia que hay entre el estado actual del sistema y el estado ideal, es decir el que debe tener. Debe saber como disminuir esa diferencia

9. Entiende y sabe moverse en el entorno político de la empresa.

Sus decisiones pueden afectar no solo al area de Ti sino muchos departamentos de la empresa, entiende la empresa y el contexto en el que esta para dar el impacto correcto en la organización

10. Le da mentoría a su equipo.

Es capaz de trasmitir sus conocimientos para que sus personas a cargo mejoren sus capacidades de diseño.

11. El arquitecto de software desarrolla

Así no sea mucho se involucra con el código para entender las consecuencias de lo que esta creando.

Cuales el el patrón correcto para un sistema

Para saberlo se debe entender lo que son:

Los atributos de calidad o requisitos no funcionales

Se hace una comparación de cada patron contra cada atributo de calidad

Escalabilidad= Es la capacidad del sistema para que con el tiempo pueda soportar cargas de trabajo mas grandes. Como por ejemplo mas usuarios

Microservicios

Arquitectura orientada a eventos

Desplegabilidad= Es la facilidad de desplegar un sistema en producción

Microkernel

Microservicios - Masomenos

Eventos - Masomenos

Capas - La peor

Rendimiento

Es cuando la aplicación tiene tiempos de respuesta apropiados a las operaciones que está haciendo

Eventos

Microkernel - regular

Capas - regular

Microservicios- Malo

Agilidad

Es que tan rapido se puede llevar a producción cambios para que la empresa se mantenga comptetitiva.

Microkernel

Eventos

Microservicios

Capas - Malas

Testeabiidad = Es que tan facil se pueden hacer pruebas en el sistema.

Capas

Microkernel

Microservicios

Eventos-Mala

Facilidad de desarollo= Es qué tan fácil es empezar a implementar el patrón de arquitectura.

Capas

Microservicios-Depende

Microkernel - Mala

Eventos – Mala

Arquitectura por capas

Es un patrón que contiene grupos de subtareas

Analogía



Si tiene que pasar SI O SI atraves de una capa para llegar a la capa de abajo, las capas no se pueden saltar. La app se separa en grupos o capas que da la estructura general

Carateristicas

-Los componentes estan organizados en capaz horizontales

-Cada capa tiene un rol y una responsabilidad unica dentro de la app

-Capas de alto y bajo nivel, alto nivel significa que esta mucho mas cerca del cliente, bajo nivel significa que es una capa mucho mas cerca de manejar cosas como hardware.

-El patrón en ningun momento define cuantas capas son ni de que tipo deben ser, se puede tener una, dos tres, seis o 10 capas en función del contexto y del sentido que tenga tenga n cantidad de capas.

-La capa de alto nivel deberia poder usar servicios de una capa de bajo nivel, pero al reves no se puede, la capa inferior no debe tener ningun conocimiento de la capa superior

-Las capas deberian tener una abstracción o encapsulamiento de la funcionalidad que tiene, algunos detalles probablemente o funciones quizas no sea conveniente que se puede acceder o interactuar con ellos. Dado que si esto pasa se pueden generar dependencias entre capas mas alla de lo necesario

-Las capaz deben estar bien definidas de tal manera que se puedan remplazar por otras capas si es necesario por algun motivo. Si se remplaza una capa a por b c deberia ser indiferente a esto.

-Los componente de cada capa solo deben hacer tareas relacionadas con la capa.

PATRÓN DE ARQUITECTURA DE TRES CAPAS

Se compone de

PRESENTACIÓN

Esta capa se encarga de manjar la interacción general entre un cliente y una aplicación, es ese punto por donde todo inicia.

El cliente puede ser:

Un ser humano

Un programa externo

Responsabilidades

-Entrega y solicita información al usuario, todo eso con el fin de alimentar las capas posteriores

-Interpreta solicitudes del usuario en acciones para las capas de negocio y persistencia

Ejemplos de componentes en la capa de presentación

-Una aplicación web

-Una interfaz por consola

-Una aplicación de escrtiorio

-Una aplicación móvil

-Un servicio web

-Un API

CAPA DE DOMINIO

-Es la que contiene la lógica de negocio del sistema, es una funcionalidad que es central para que el negocio opera y esta existe haya o no sistema de información.

-Contiene el trabajao o las tareas para las cuales el sistema esta hecho.

-Normalmente, la capa de dominio deberia esconder el acceso a datos.

Ejemplos

-Calculos basados en datos de entrada y datos almacenados, como calcular un impuesto, un salario, una venta, es decir toma los datos de la capa d epresentación o de acceso a datos y hace calculos.

-Validaciones de datos que vienen de la capa de presentación

-Definir que algoritmo o logica de negocio utilizar

Otros nombres para la capa de dominio

-Capa de negocio

-Capa de lógica de negocio

-Capa de lógica de dominio

CAPA DEACCESO A DATOS

-Se comunica con bases de datos u otros sistema de perseistencia para obetener y guarar información, es decir, es la que tiene el codigo para obtener los datos de las fuentes d einformación como la base de datos y las diversas opciones que existen

-De forma mas general, se comunica con sustema que realizan tareas en n nombre de la aplicación

-Monitores de transacciones

-Sistemas de mensajeria

-Entre otros

Otros nombres para la capa de acceso a datos.

-Capa de persistencia

-Capa de datos

PATRÓN DE ARQUITECTURA MICROSERVICIOS

Es una patrón de arquitctura considerado mas de moda o mas moderno.

Generalidad basicas

Es muy importante aprender que son las aplicaciones monoliticas, lo mas común al crear una aplicación es tener una aplicación que tenga un cliente y un servidor. Eso esta junto en una sola app ejecutable, esto se ve como si fuera como si fuera un gran bloque junto, por eso se le llama un monolito, escalarlas es dificl y caro otro reto que hay es que para desplegar un cambio hay que desplegar la aplicación entera

De aquí es que surge la idea o lo que viene a romper los microservicios, Esta habla de desarrollar una aplicación como un conjunto de servicios pequeños, en lugar de un gran monolito donde esta toda la app, ese monolito se separa en servicios mucho mas pequeños con las siguientes caracteristicas

-Cada microservicio corre su propio proceso, es decir son procesos totalmente independitenes y no hay como un solapamiento del uno con el otro.

-Si los microservicios necesitan comunicarse, lo hacen con mecanismos muy livianos

-Se desarrollan alrededor con las necesidades del negocios, hay una relación muy fuerte entre la forma en como opera el negocio y la forma en como se desarrollan los microservicios.

-Tienen que estar soportados por un proceso de despligue automatico.

Tpologías -Distintas formas en las que los servicios pueden estar oganizados

1. Topología basada en APIS Rest, Se tiene un conjunto de servicios y estos al ser consumidos no son accedidos directamente si no que se usa una capa de API y esta capa de API y es la que expone las URLS publicas que los usuarios van a acceder, entonces la idea es que si un usuario invoca una de estas URLS, la capa de API lo que hace redireccionar la URL hacia el microservicio o hacia el componente de servicio apropiado y ese servicio la genera y procesa y envia de regreso para que la API le responda al cliente.
2. Topología basa en aplicación, la aplicación que accede a los microservicios normalmente es una aplicación hecha por le desarrollador y esta APP lo que hace es conectarse directamente a los microservicios, los cliente interactuan con esa app y esta es la que encapasula o esconde los llamados a los microservicios. Se usa generalmente en apps pequeñas o medianas.
3. Topologías basadas en mensajeria, en este caso no se interactuan con los microservicios atraves de REST si no que se hace atraves d eun broker de mensajeria, normlamente en estos casos,hay una aplicación que es la qu e se encarga de acceder la cola y cada vez que los usuarios hacen solicitud a la app, esta pone un mensaje en la cola, y la cola lo que hace es redigiri ese mensaje al mensaje servicio correspondiente y luego devolver la respuesta, este es un enfoque mas usado en aplicaicón grandes donde se suele necesitar mas ocntrol en la capa de transporte.

ORIENTADA A EVENTOS

Componentes

-Generadores o productores.

Se encargan de generar los distintos eventos del sistema.

Mensajes, son los que contienen la información del evento, es como la pieza que contiene la información relacionada con el evento.

Componente de mensajeria, es la platafroma que recibe el mensaje hace un procesamiento con el mensaje si es necesario e informar a las parte interesadas es decir los distintos consumidores.

Canales, son como los canales entre el componente de mensajeria y los componentes interesados en ese notificados acerca del evento. Ya sea con colas de mensajes – Patron publicador - subscritor

Procesadores o consumidores, son los componentes interesados en los eventos, tienen una unica tarea relaiconada con el negocio, que sea lo mas espesficia posible, principios SOLID.

Dben estar lo mas desacoplados posibles de otras partes de la APP, no deberian tener nigun conocimiento de quienes generan los eventos, ni de otros procesadores que puedan existir en el sistema, lo ideal es que sean lo mas independientes posible de las otras partes de la arquitetcura del sistema.

Topologías, son basicamente la forma de organziarel componente de mensajeria

El mediador y el broker

En el mediadior la idea basica es que dentro de el componente de mensajaria el mediador se encarga de sincronizae to-dos los procesadores, se encrga de que los procesadores se ejecuten en la secuencia correcta para producir el resultado deseado.

MVC –Modelo vista controlado

Vista

-Muestra el modelo

-Múltiples vistas para un modelo

-Actualiza con notificaciones del modelo

Controlador

-Lógica de aplicación

-Recibe solicitudes de la vista

-Ejecuta acciones en el modelo

Modelo

-Lógica de negocio

-Tiene datos, comportamiento y estado

-Independiente de controlador y vista

-Activo o pasivo

¿MVC es lo mismo que una arquitectura de tres capas?

Es similar, si se hiciese una relación podría ser así

Vista - Presentación

Controlador – Dominio = Aquí si hay una gran diferencia

Modelo – Acceso a datos = Hay similitud pero hay diferencis muy importantes