1. Lectura previa

1.1 Leer los capítulos introductorios de Clean Architecture (Robert C. Martin).

1.2 Subrayar y anotar ideas clave relacionadas

¿Qué es arquitectura de software?

Es la estructura fundamental de un sistema, la forma en que están organizados sus componentes y cómo se comunican entre sí

No solo se refiere a diagramas o modelos sino a las decisiones estratégicas sobre diseño que impactan en la vida útil del software

Su propósito principal es maximizar la vida útil del sistema y facilitar su evolución

¿Por qué define el éxito o fracaso de un producto?

Una buena arquitectura permite que el software sea fácil de cambiar, mantener y escalar con el tiempo

Una mala arquitectura genera un sistema rígido que se vuelve costoso y riesgoso de modificar lo cual puede llevar al fracaso del producto

El código fuente cambia constantemente pero la arquitectura es lo que garantiza que esos cambios no rompan todo el sistema

O sea que:

Éxito = Arquitectura adaptable, que permite responder rápido al mercado y a nuevas necesidades

Fracaso = Arquitectura rígida que vuelve lento y caro el desarrollo

Principales atributos de calidad: escalabilidad, seguridad, disponibilidad, mantenibilidad, flexibilidad

Escalabilidad

El sistema debe poder crecer en usuarios datos o funcionalidades sin perder rendimiento

Seguridad

Proteger los datos y las interacciones del sistema contra accesos no autorizados o ataques

La arquitectura define cómo se autentican los usuarios cómo se encriptan datos y cómo se segmenta la información

Disponibilidad

El sistema debe estar accesible la mayor parte del tiempo.

La arquitectura influye en la redundancia tolerancia a fallos y planes de recuperación

Mantenibilidad

Capacidad de modificar y corregir el sistema de manera rápida y segura

Un diseño limpio y desacoplado permite arreglar errores y añadir funciones sin romper lo existente

Flexibilidad

Facilidad para adaptarse a cambios en los requisitos del negocio o en la tecnología

Una arquitectura flexible evita el “lock-in” con un proveedor o framework

2. Preparación del debate (individual):

2.1 Redactar un resumen de máximo 1 página:

La arquitectura de software puede entenderse como el conjunto de decisiones que definen la estructura de un sistema y la forma en que sus diferentes elementos se relacionan entre sí Desde una perspectiva personal no se trata únicamente de diseñar cómo se verán las partes de un sistema o de elegir qué tecnologías se van a emplear sino de crear una base sólida que garantice que el producto pueda crecer mantenerse y adaptarse a las necesidades del entorno con el paso del tiempo Una buena arquitectura es como los cimientos de una construcción pues aunque no se ven son los que determinan la resistencia la estabilidad y la vida útil de todo lo que se construya encima

Cuando la arquitectura de un sistema está bien diseñada el software se vuelve más sencillo de entender de modificar y de extender lo que a la larga se traduce en un producto más competitivo y con mayores posibilidades de éxito En cambio cuando las decisiones arquitectónicas se toman sin una visión a largo plazo se generan sistemas frágiles difíciles de mantener y costosos de escalar lo que puede llevar incluso al fracaso de un proyecto o de una empresa que dependa de él Por eso se puede decir que la arquitectura de software es uno de los factores que más influyen en el éxito o fracaso de un producto digital

Un atributo esencial de la arquitectura de software es la escalabilidad La escalabilidad permite que un sistema que empieza siendo pequeño pueda crecer junto con la cantidad de usuarios de datos y de procesos sin perder velocidad ni calidad de servicio Este atributo es muy importante para proyectos que esperan un crecimiento acelerado porque asegura que no sea necesario rehacer todo el sistema desde cero cuando se multiplique la demanda

Otro atributo fundamental es la seguridad La seguridad no se limita a poner contraseñas fuertes sino a diseñar el sistema de manera que proteja los datos sensibles evite accesos indebidos y reduzca los riesgos frente a ataques externos Cuando un sistema es inseguro los usuarios pierden la confianza y la empresa corre grandes riesgos legales económicos y de reputación Por eso la seguridad se considera uno de los pilares de una arquitectura sólida

Un tercer atributo clave es la mantenibilidad La mantenibilidad significa que el sistema se puede modificar corregir y mejorar sin que eso represente un riesgo de romper otras partes ni un gasto excesivo en tiempo o dinero Este atributo permite que los equipos de desarrollo respondan rápido a cambios en el mercado a nuevas necesidades de los usuarios o a la corrección de errores inevitables Con una buena mantenibilidad el software puede evolucionar y seguir siendo útil durante años mientras que sin ella se vuelve rígido y termina quedando obsoleto muy pronto

2.2 Pregunta para el debate:

Qué debería priorizar una empresa nueva: escalabilidad rápida o facilidad de mantenimiento?

Ideas subrayadas

Cuando hablamos de arquitectura de software, el software es de naturaleza recursiva y fractal, grabado y esbozado en código. Todo son detalles. Niveles entrelazados de

No se necesita una gran cantidad de conocimientos y habilidades para que un programa funcione. Los niños de secundaria lo hacen todo el tiempo. Hombres y mujeres jóvenes en la universidad inician negocios de miles de millones de dólares basándose en unas pocas líneas de PHP o Ruby. Hordas de programadores jóvenes en granjas de cubos de todo el mundo revisan enormes documentos de requisitos almacenados en enormes sistemas de seguimiento de problemas para lograr que sus sistemas "funcionen" mediante la pura fuerza bruta de la voluntad. El código que producen puede no ser bonito; pero funciona. Funciona porque hacer que algo funcione (una vez) no es tan difícil.

Hacerlo bien es otra cuestión completamente diferente. Obtener el software correcto es difícil. Se necesitan conocimientos y habilidades que la mayoría de los programadores jóvenes aún no han adquirido. Requiere pensamiento y conocimiento que la mayoría de los programadores no se toman el tiempo para desarrollar. Requiere un nivel de disciplina y dedicación que la mayoría de los programadores nunca imaginaron que necesitarían. Principalmente, se necesita pasión por el oficio y el deseo de ser un profesional.

Uno de los objetivos de este libro es romper con toda esa confusión y definir, de una vez por todas, qué son el diseño y la arquitectura. Para empezar, afirmaré que no hay diferencia entre ellos. Ninguno en absoluto.

La palabra "arquitectura" se usa a menudo en el contexto de algo de alto nivel que está divorciado de los detalles de nivel inferior, mientras que "diseño" parece implicar más a menudo estructuras y decisiones de un nivel inferior. Pero este uso no tiene sentido.

El objetivo de la arquitectura de software es minimizar los recursos humanos necesarios para construir y mantener el sistema requerido.

Pero una parte de su cerebro sí duerme: la parte que sabe que un código bueno, limpio y bien diseñado es importante.

Estos desarrolladores creen en una mentira familiar: "Podemos limpiarlo más tarde; ¡Solo tenemos que llegar al mercado primero! Por supuesto, las cosas nunca se arreglan después, porque las presiones del mercado nunca disminuyen. Llegar primero al mercado simplemente significa que ahora tienes una horda de competidores detrás de ti y tienes que adelantarte a ellos corriendo lo más rápido que puedas.

Y por eso los desarrolladores nunca cambian de modo. No pueden regresar y limpiar las cosas porque tienen que terminar la siguiente función, y la siguiente, y la siguiente, y la siguiente. Y así el desorden crece y la productividad continúa su aproximación asintótica hacia cero.

Así como la Liebre confiaba demasiado en su velocidad, los desarrolladores confían demasiado en su capacidad para seguir siendo productivos. Pero el creciente desorden de código que mina su productividad nunca duerme y nunca cede. Si se le permite, reducirá la productividad a cero en cuestión de meses.

La mentira más grande que creen los desarrolladores es la noción de que escribir código desordenado los hace ir más rápido en el corto plazo y solo los ralentiza a largo plazo.

En todos los casos, la mejor opción es que la organización de desarrollo reconozca y evite su propio exceso de confianza y comience a tomar en serio la calidad de su arquitectura de software.

El primer valor del software es su comportamiento. Se contratan programadores para hacer

El software fue inventado para ser "suave". Su objetivo era ser una forma de cambiar fácilmente el comportamiento de las máquinas. Si hubiéramos querido que el comportamiento de las máquinas fuera difícil de cambiar, lo habríamos llamado hardware.

Para cumplir su propósito, el software debe ser blando, es decir, debe ser fácil de cambiar. Cuando las partes interesadas cambian de opinión sobre una característica, ese cambio debería ser simple y fácil de realizar. La dificultad para realizar tal cambio debería ser proporcional sólo al alcance del cambio y no a la forma del cambio.

- Si me das un programa que funciona perfectamente pero que es imposible de cambiar, entonces no funcionará cuando cambien los requisitos y no podré hacerlo funcionar. Por lo tanto el programa

El segundo valor del software, la arquitectura, es importante pero nunca particularmente urgente.

Sólo recuerde: si la arquitectura es lo último, entonces el desarrollo del sistema será cada vez más costoso y, eventualmente, el cambio será prácticamente imposible para una parte o la totalidad del sistema. Si se permite que eso suceda, significa que el equipo de desarrollo de software no luchó lo suficiente por lo que sabían que era necesario.