12. Arquitectura de un sistema

Definición

> *IEEE-1471*
>
> - Propiedades fundamentales de un sistema en su entorno encarnado en sus elementos, relaciones y en los principios de su diseño y evolución.
> - La arquitectura de software representa la estructura o las estructuras del sistema, que consta de componentes de software, las propiedades visibles externamente y las relaciones entre ellas.

> *Martin Fowler*: son aquellas decisiones que son importantes y difíciles de cambiar.

La arquitectura de un sistema no solo se trata de sus componentes, también de la conexión entre estos y las decisiones tomadas y por tomar para construirlo y hacerlo evolucionar. Puede ser

- la forma del sistema
- las decisiones significativas de diseño
- se define por el costo de cambiar algo de este diseño
- es la división del sistema en componentes
- representa la comunicación entre componentes

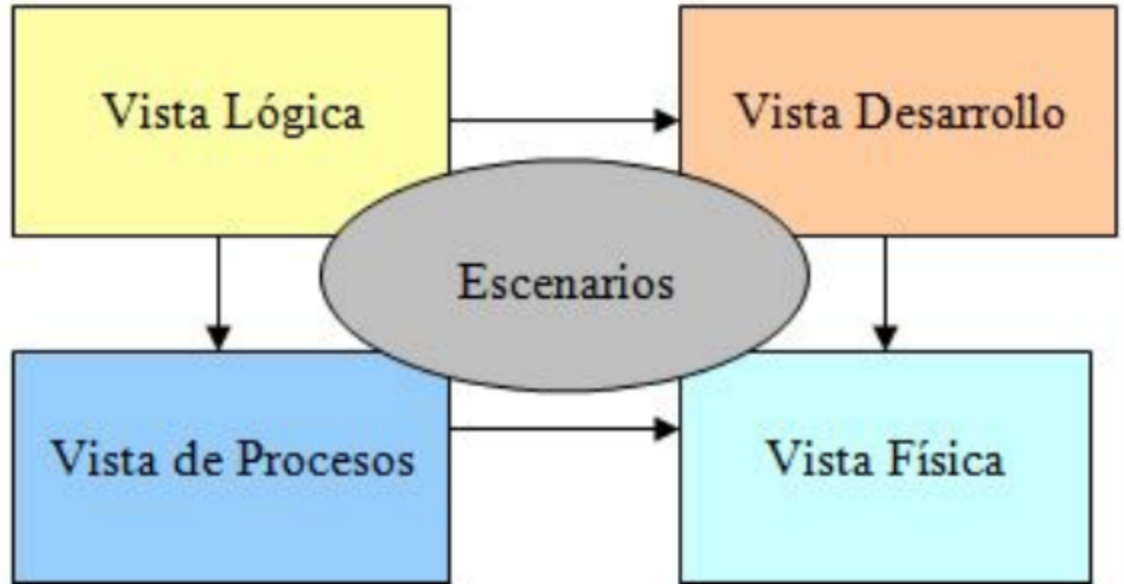
documentar una arquitectura

- necesitamos dirigir la documentación a todos los stakeholder del producto, cada uno se interese da algo diferentes de nuestro sistema, debemos de satisfacer sus necesidades en nuestra documentación. Dentro de estos encontramos:

- clientes
- usuarios
- product owners
- UX Researcher, UI Designer, UX Writer
- Scrum masters
- Desarrolladores
- Testers
- Administradores de bases de datos
- Devops, gente de infraestructura, de despliegue

Se busca un único documento o diagrama que eplique la arquitectura, para ello se propone el modelo de vistas 4+1 - Kruchten.

Modelo de vistas 4+1



A través de diferentes vistas, analizamos diferentes perspectivas del problema en cuestión. Como resultado, en un *único documento concentramos las principales decisiones tomadas sobre el sistema*

Con este documento logramos que:

- nuevos integrantes del equipo puedan entender la arquitectura del sistema y que puedan ubicarse dentro de la solución

- discutir diferentes decisiones con todos los stakeholders y obtener una validación temprana.
  véase Ejemplo Modelo 4+1

Vista lógica

- requisitos funcionales (lo que el sistema debe brindar en términos de feautures al usuario)
- se aplican *principios de abstracción, encapsulamiento y herencia.*
- de esta forma identificamos mecanismos y elementos de diseño *comunes* a diversas partes del sistema.
- de *interés para los desarrolladores*.

Vista de desarrollo o de componentes

- se refiere a la organización real de los módulos de software en el ambiente de desarrollo.
- el software se empaqueta en partes pequeñas (bibliotecas de programas o subsitemas) que pueden ser desarrollados.
- la vida de desarrollo tiene en cuenta los *requisitos relativos a la facilidad de desarrollo*, administración del software, reutilización de elementos comunes, restricciones impuestas por las herramientas o el lenguaje usado.
- si es que se organizaron los paquetes basados en features, también sirve de documentación de los features entregados.

Vista de procesos

- requisitos no funcionales (rendimiento, disponibilidad y concurrencia). no se trata de lo que hace el sistema, sino de cómo lo hace en términos de resiliencia y eficiencia.
- se enfoca en el performance y la disponibilidad: asuntos de concurrencia y distribución, integridad del sistema y tolerancia a fallas.

Procesos & vista de procesos

- Un proceso es una agrupación de tareas que forman una unidad ejecutable.
- representan el *nivel de control táctico de la vista de procesoa*. Es decir, se puede comenzar, recuperar, reconfigurar y detener.
- pueden replicarse para *aumentar la distribución de la carga de procesamento* (o para *mejorar la disponibilidad*)

Vista de física o de despliegue

- *requisitos no funcinoales del sistema*:
    - disponibilidad (¿es siempre accesible?)
    - confiabilidad (tolerancia a fallas, digamos de un nodo)
    - rendiminto (cuántas operaciones por segundo puede manejar?)
    - escalabilidad (puede escalar sin perder rendimiento?)
- Los elementos tales como *redes, procesos, tareas* (por ejemplo la autenticación) y *objetos*, requieren ser mapeados sobre los nodos. La vista física muestra la **distribución en la red de hardware**, **cómo se instala y ejecuta** el software en estos nodos.
- Se debe usar diferentes configuraciones: algunas para *desarrollo y pruebas*, otras para *mostrar el sistema en diferentes dispositivos* para diferentes usuarios. (por esto buscamos que la relación del software en los nodos sea flexible y que minimice el impacto en código fuente)

Vista de *Escenarios*

Los escenarios representan una abstracción de los requisitos más importantes (casos de uso).
Los elementos de las *4 vistas* trabajan conjuntamente mediante el *uso de un conjunto de escenarios relevantes*.
La vista de escenarios sirve a los propósitos:

- guía para *descubrir elementos arquitectónicos* durante el diseño de arquitectura.

- punto de partida para las *pruebas que el prototipo de la arquitectura debe cumplir*.
- rol de validación e ilustración después de completar el diseño de la arquitectura (ver *si el sistema cumple con los casos de uso planteados*).

Conclusiones 4+1

- A través de diferentes vistas analizamos diferentes perspectivas del problema en particular
- Concentra en un único documento las principales desciones tomadas en el sistema
- Permite que nuevos integrantes del equipo entiendan la arquitectura y puedan ubicarse en el dominio de la solución de buscan implementar
- Permite dicutir con todos los stakeholder, validar decisiones tempranamente.