

FUNDAMENTOS DE ARQUITECTURA DE SOFTWARE

En la arquitectura del software se habla de:

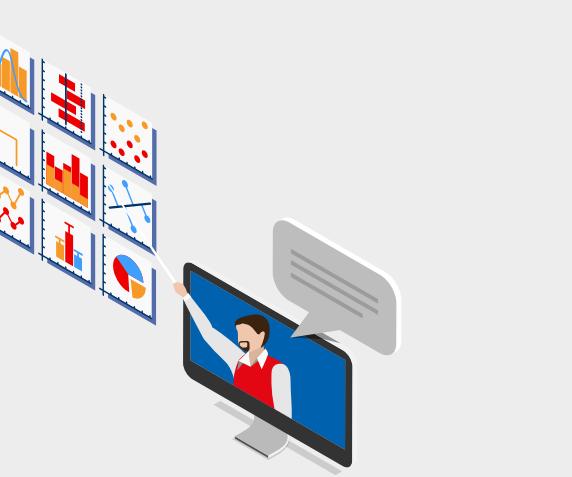
- Estructuras.
- Modelos con diagramas.
- Comunicación entre diferentes sistemas.



Atravesar todo el camino para atender que es el proceso de desarrollo y como la arquitectura está involucrada en cada uno de los pasos del proceso de desarrollo del software.



Cuál es el rol del arquitecto y como el arquitecto puede ayudar al éxito o fracaso de un sistema.



Hacer evidente decisiones que a veces son implícitas y nos va ayudar a ser consciente de cuando estamos tomando una decisión arquitectónica en un sistema y cómo hacer para tomar la mejor decisión posible en ese momento.



¿QUE ES ARQUITECTURA DE SOFTWARE?

También denominada **arquitectura lógica**, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan un marco definido y claro para interactuar con el código fuente del software.

Modela elementos de software, sus propiedades visibles. Es considerada un conjunto de descripciones principales de diseño para llegar a un resultado de calidad en el sistema. La arquitectura puede emerger de un equipo autogestionado o de un arquitecto externo, y está claramente influenciada según el marco de referencia en el que se trabaje.



Estilo: Centradas en datos

Pizarrón:

El pizarrón es el núcleo de la arquitectura. Donde componentes externos a él se encargarán de procesar un dato y escribirlo en el pizarrón (Este funciona como centralizador). Cuando el pizarrón ya tiene todos los datos necesarios, el mismo podría generar una salida. Ejemplo: Sistema Fiscales

Centrado en base de datos:

Es un estilo común; Se trata de que una cantidad de componentes comparte una misma base de datos. De Ejemplo: aplicaciones que poseen comunicación por Internet.

Sistema experto Basado en reglas:

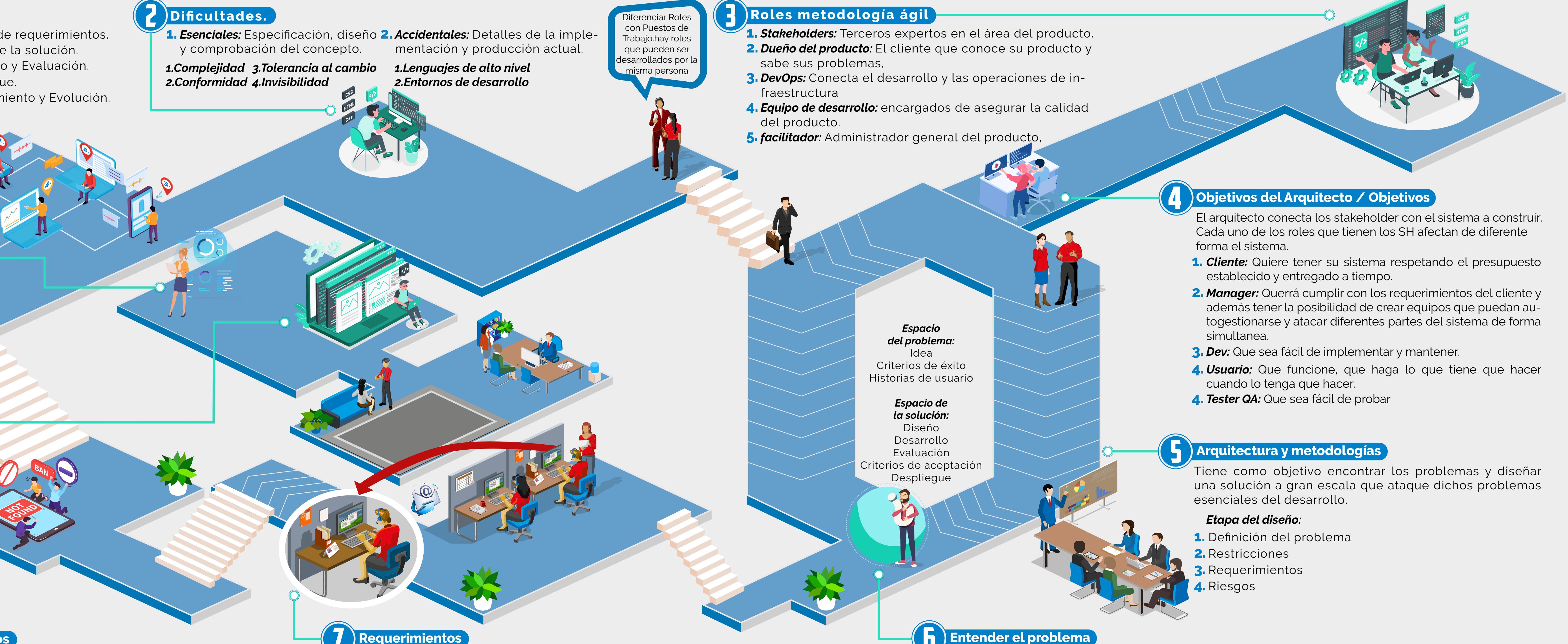
Este sistema no se ve muy seguido en aplicaciones modernas, un componente A (Tipo Cliente) consulta a uno B, donde este se encargará de tratar de entender si la petición del cliente es una consulta o regla. Para que el componente B logre resolver la petición se va a comunicar con un tercer componente.

El ciclo de vida del producto:

agregará limitaciones al producto, por ejemplo a medida que avanza el proceso de implementación el modelo de datos va a ser más difícil de modificar.

1 Etapas

1. Análisis de requerimientos.
2. Diseño de la solución.
3. Desarrollo y Evaluación.
4. Despliegue.
5. Mantenimiento y Evolución.



2 Dificultades.

1. **Eenciales:** Especificación, diseño y comprobación del concepto.
2. **Accidentales:** Detalles de la implementación y producción actual.
3. **Complejidad**
4. **Tolerancia al cambio**
5. **Mantenimiento y Evolución.**

3 Roles metodología ágil

1. **Stakeholders:** Terceros expertos en el área del producto.
2. **Dueño del producto:** El cliente que conoce su producto y sabe sus problemas.
3. **DevOps:** Conecta el desarrollo y las operaciones de infraestructura
4. **Equipo de desarrollo:** encargados de asegurar la calidad del producto.
5. **facilitador:** Administrador general del producto.

4 Objetivos del Arquitecto / Objetivos

El arquitecto conecta los stakeholder con el sistema a construir. Cada uno de los roles que tienen los SH afectan de diferente forma el sistema.

1. **Cliente:** Quiere tener su sistema respetando el presupuesto establecido y entregado a tiempo.
2. **Manager:** Querrá cumplir con los requerimientos del cliente y además tener la posibilidad de crear equipos que puedan autogestionarse y atacar diferentes partes del sistema de forma simultanea.
3. **Dev:** Que sea fácil de implementar y mantener.
4. **Usuario:** Que funcione, que haga lo que tiene que hacer cuando lo tenga que hacer.
4. **Tester QA:** Que sea fácil de probar

5 Arquitectura y metodologías

Tiene como objetivo encontrar los problemas y diseñar una solución a gran escala que ataque dichos problemas esenciales del desarrollo.

Etapa del diseño:

1. Definición del problema
2. Restricciones
3. Requerimientos
4. Riesgos

6 Entender el problema

Lo más importante es separar la comprensión del problema de la propuesta de solución. Cuando no separamos esto, tendemos a ver cómo parte del problema ciertas cuestiones tecnológicas, cuando en realidad son detalles de implementación.

Solución: Brinda el detalle del 'cómo' se va a resolver?, reflejando los detalles del problema detectado y evitando resolver problemas que no se quiere o necesita resolver y consta de 3:

1. **Desarrollo:** escribir el código, configuraciones y contrataciones de servicios.
2. **Evaluación:** medir la eficiencia y eficacia del software frente al problema.

Medir el impacto del software, no importa lo bueno que sea el problema si los usuarios no lo usan o no le ven uso.

7 Requerimientos

Una vez que entendemos el espacio del problema y el espacio de la solución, vamos a entrar a analizar los requerimientos de nuestro sistema.

Identificamos riesgos funcionales: Se calificará su riesgo de acuerdo a su dificultad o complejidad.

Atributos de calidad NO funcionales: Se calificará su riesgo de acuerdo a la incertidumbre que genere, cuanto mas incertidumbre hay, mas alto es el riesgo.

Capa Funcional: se ven alimentados por requerimientos del sistema, ¿qué cosas tienen que pasar operativamente?

Esta capa se ve afectada por las restricciones que pueden afectar operativamente a lo funcional.

8 Riesgos

Es importante cubrir los riesgos tratando de describir que pasaría si no llegáramos a cubrir ese riesgo?

Capa de requerimientos de negocio: Reglas del negocio que alimentan los requerimientos del negocio.

Capa de usuario: Cómo el usuario se desenvuelve usando el sistema,

qué atributos del sistema se deben poner por encima de otros.

Conocimiento del dominio: Riesgo prototípico, son aquellos que podemos atacar de forma estándar.