3. Arquitectura del Software

Introducción

La arquitectura de software tiene como objetivo identificar los subsistemas dentro de un sistema y diseñar la forma que interactúan entre ellos. **Es la estructura de un sistema que comprende los elementos del software, las propiedades externamente visibles de esos elementos y la relación entre ellas.**

En esta parte del diseño de software, tenemos el **diseño de más alto nivel**, en donde se hacen

- Elecciones de tecnología
- Decisiones sobre productos a usar, servidores

Aquí, al trabajar, se divide al sistema en partes lógicas tal que cada una puede ser **comprendida independientemente**, y describe las relaciones entre ellas.

Rol

Algunos de los roles que le dan sentido a esta parte del diseño de software son:

- Comprensión y comunicación: la arquitectura debe **facilitar la comunicación** entre partes a través del uso de un marco de compresión común entre los interesados que oculta la complejidad de cada subsistema.

- Reuso: hacer a los **componentes** del sistema capaces de ser reutilizados o **flexibles** ante cambios en la arquitectura.
- Construcción y evolución: la arquitectura servirá para **brindar una estructura en el cual el desarrollo del sistema se pueda construir y usar de forma ordenada y correcta**. Además, al estar compuesta de varios componentes, permite el trabajo enfocado individual y simultáneo para ayudar a la evolución del software.
- Análisis: gracias a una arquitectura, podemos usar **estimadores** para poder analizar el sistema en distintos contextos. Es decir, la arquitectura debe tener una forma de poder, a través de un análisis, “predecir” su comportamiento. (En otras palabras, su composición debe estar tan bien organizada que pueda ser vista a futuro en contextos deseados para su posterior análisis).

> 💡 No hay regla general sobre cómo debe estar organizado un sistema o sobre qué arquitectura se debe tomar, eso depende de lo que se esté buscando hacer.

Vistas

Las vistas en arquitectura se refiere a perspectivas o enfoques específicos que se usan para analizar y comprender distintos aspectos de un sistema de software complejo.

En arquitectura, tenemos distintas vistas comprendidas de componentes relacionados entre sí, específicas a lo que se quiera analizar.

En ingeniería civil, por ejemplo, existen distintas “vistas” sobre la arquitectura de una casa. Por ejemplo, hay planos que describen la instalación eléctrica de la casa, otro plano para ver las instalaciones de gas, etc. Cada vista se **centra en un aspecto particular de un sistema** para poder **exponer propiedades del mismo**. Esto, como ya dijimos varias veces, **disminuye complejidad**.

Tres vistas comunes

1. **Módulo**: esta vista se centra en los elementos básicos de código que forman el sistema. Compuesta entonces por unidades de código que pueden contener paquetes, clases, funciones, etc. Cada módulo de código se relaciona entre sí.

2. **Componentes y conectores**: esta vista es fundamental para entender como las partes del sistema se comunican y colaboran para que el sistema total funcione bien.

   Aquí se definen componentes (entidades de ejecución que realizan funciones) que pueden estar asociadas a objetos, archivos, bibliotecas. Ejemplo, en orientado a objetos, los componentes pueden ser objetos individuales que encapsulan funcionalidades.

   Los conectores permiten la interacción y comunicación entre los componentes. Pueden ser pipes y sockets, memoria compartida, protocolos.

   Suele ser la más usada.

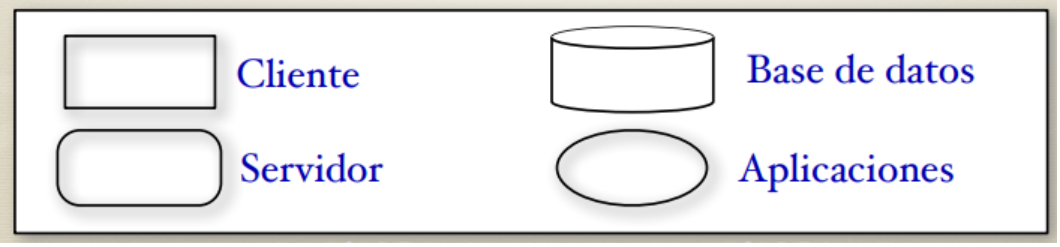
3. **Asignación de recursos**: se centra en cómo los elementos del software se relacionan y se asignan a los recursos del sistema, como hardware, sistemas de archivos, personal, entre otros. Exponen propiedades estructurales como qué proceso ejecuta en qué procesador, qué archivo reside dónde, etc.

Nos enfocaremos en Componentes y conectores (C&C)

Componente y conectores

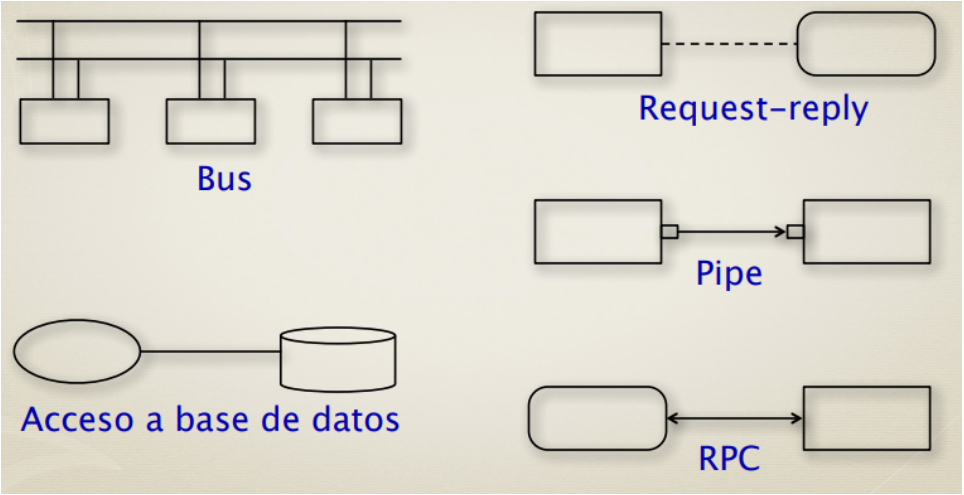
Esta contiene dos elementos principales:

- **Componentes**: elementos computacionales o de almacenamiento de datos. Tienen nombres, roles y tipos representados por símbolos (cuadrado, rectangular, circular, etc). Utilizan interfaces o puertos para su comunicación.



NO se toman las formitas

- **Conectores**: son mecanismos de interacción entre las componentes, básicamente describen el medio de interacción. Algunos ejemplos son los puertos TCP/IP, protocolos como HTTP, etc. Tienen nombres y tipo. A veces pueden representar protocolos.

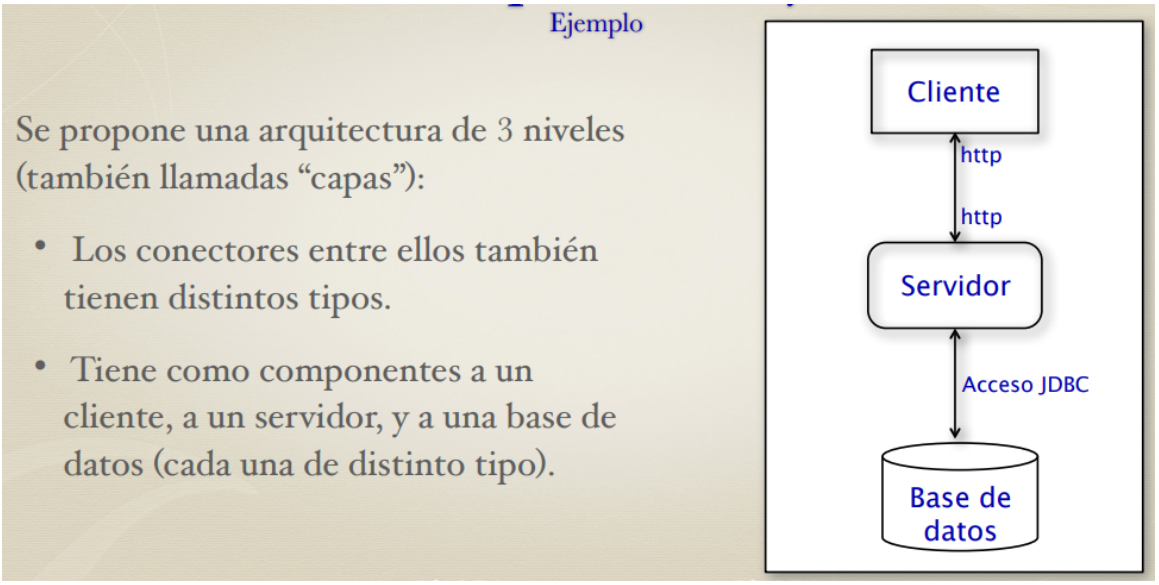


Tipos de conectores, no se toman las formitas

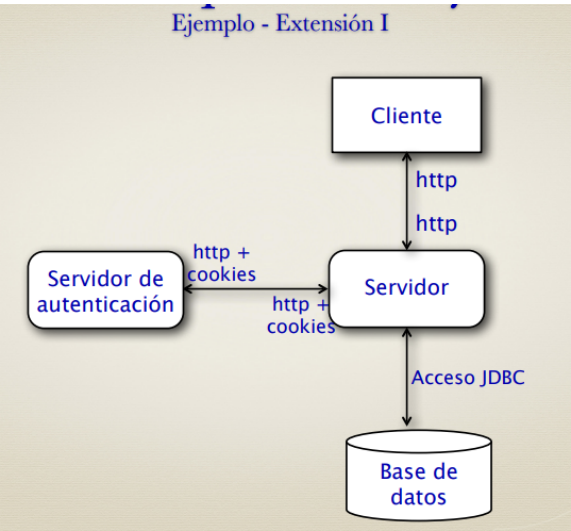
Esta vista va a definir los componentes y cómo serán las conexiones en **tiempo de ejecución** representado por gráficos.

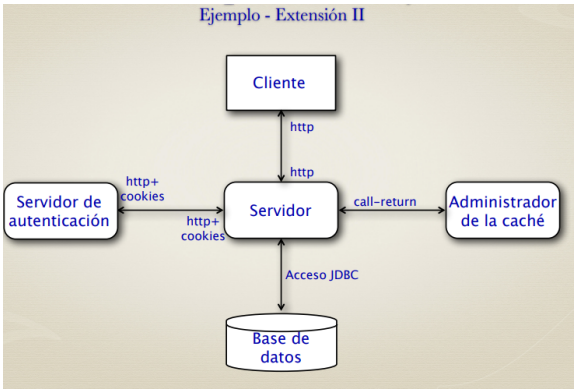
Ejemplo:

Se quiere una arquitectura de un sistema de encuesta a alumnos en donde las encuestas completadas se envían y se muestran sus resultados.



Esta arquitectura además puede sufrir extensiones:



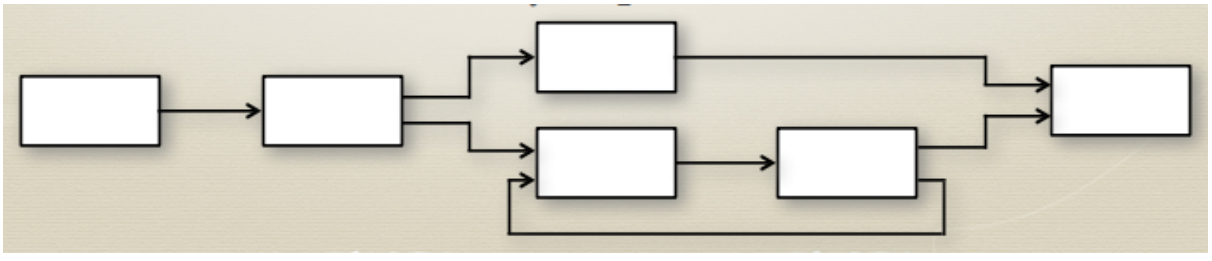


Estilos Arquitectónicos

Los "estilos arquitectónicos" son patrones o enfoques predefinidos que ofrecen soluciones para la organización y estructura de sistemas de software en la vista de componentes y conectores (C&C).

Tubos y Filtros (va completo)

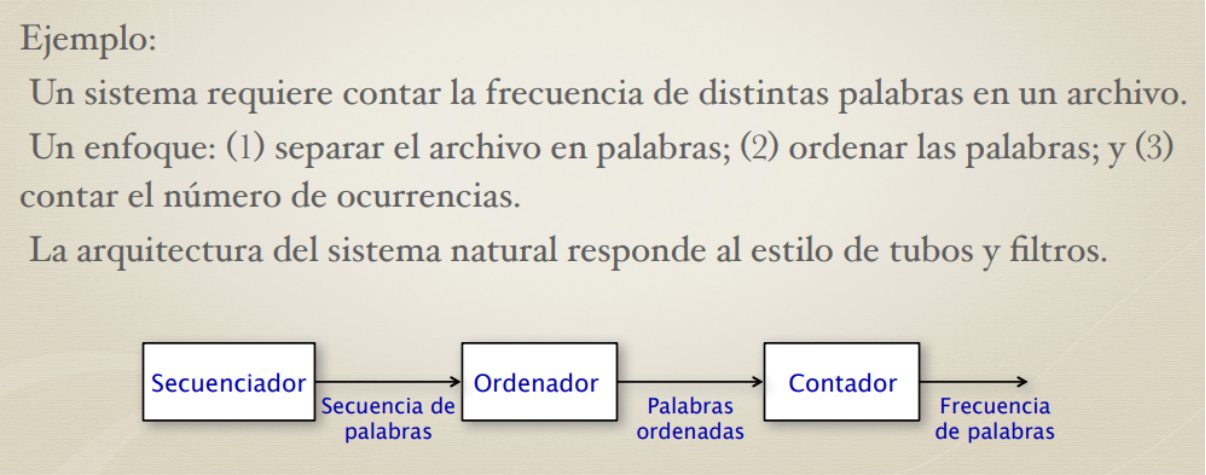
Adecuado para sistemas que hacen transformaciones de datos. El tipo de componente que tiene son los **filtros** y el tipo de conector son los **tubos**. Un filtro realiza transformaciones y el tubo le pasa los datos a otro filtro.



Características

- Los filtros son independientes y asíncronos, es decir, que tengan sus propios hilos de control (realiza transformaciones sin estar cronometrado)
- Los tubos son unidireccionales que conectan a dos componentes nomás.
- Los tubos no se bifurcan.

- Los filtros hacen buffering y sincronización para asegurar funcionamiento correcto como productor y consumidor (o sea, que no haya problemas de rendimiento, concurrencia y coordinación).



Datos Compartidos

Se centra alrededor del intercambio y acceso a datos compartidos. Este estilo se usa para sistemas de colaboración y acceso a los mismos datos.

Se tienen **repositorios de datos** que almacenan la data y **usuarios de datos** que la consultan y utilizan.

Hay un sólo tipo de **conector: lectura/escritura**.

1. Variante estilo pizarra: la modificación de un repo se informa a todos los usuarios
2. Variante estilo repositorio: los usuarios son los encargados de obtener las actualizaciones del repo, el repo solo actúa como una fuente central de datos compartida (no desencadena actualizaciones como el estilo pizarra). Orientado a sistemas de base de datos, sistemas web, etc.

Estilo de datos compartidos

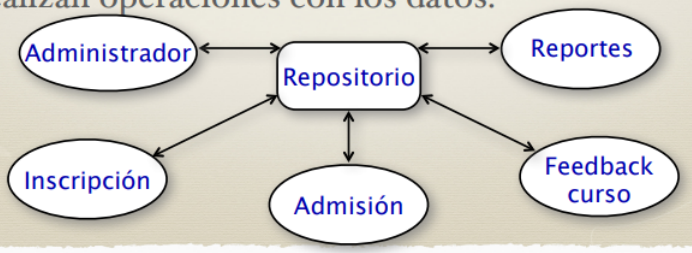
Más ideas => guaraní

Ejemplo:

Sistema de inscripción de alumnos a los cursos.

El repositorio contiene toda la información sobre los alumnos, cursos, horarios, correlatividades, etcétera.

Componentes usuarias: administrador, inscripción, admisión, reportes, etcétera; que realizan operaciones con los datos.

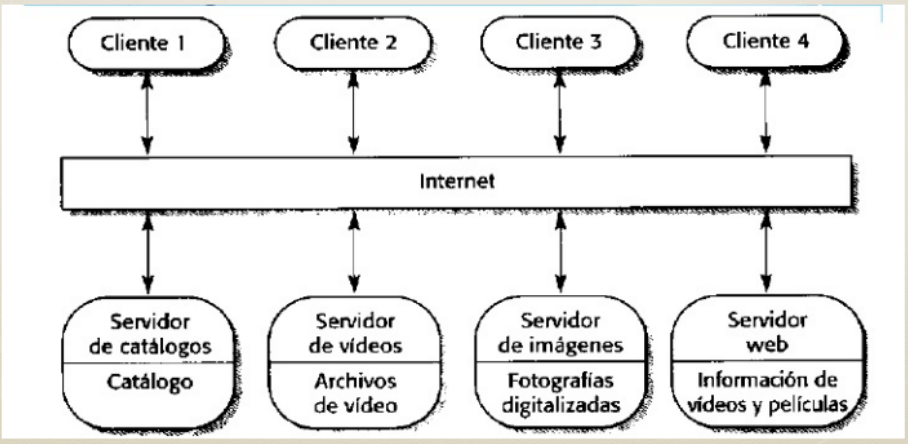


Cliente-servidor

Tenemos dos componentes (mencionados en el título) en donde:

- Clientes se encargan de iniciar y **comunicarse sólo con los servidores**
- El conector son las **requests y replies**.
- Ambos componentes **residen en máquinas distintas**.
- Su arquitectura clásica es de **tres niveles** (suma a la base de datos).

Ejemplo: Sistema de usuarios en web, de una biblioteca de imágenes



Otros estilos de la vista de Componentes y Conectores

- Publicar-suscribir (como el “foro” de famaf): dos tipos de componentes, publicar evento y suscribir.
- Peer-to-peer (único componente): único tipo de componente, cada componente le pide servicios a otro (modelo OO)
- Procesos que se comunican a través de pasajes:

Documentación en Diseño Arquitectónico

Además del uso de **diagramas** para discutir y crear diseños, se usan **documentos escritos** organizados compuestos por las secciones:

1. **Contexto del sistema y la arqui**: diagrama de contexto provee contexto general. Establece alcance del sistema, actores principales, fuentes, etc.
2. **Descripción de vistas**: se describen las vistas elegidas
    a. Presentación principal de la vista: descripción gráfica
    b. Catálogo de elementos: provee mas data sobre (a).
    c. Fundamento de la arqui: justificación de decisiones tomadas
    d. Comportamiento: idea del comportamiento real del sistema en escenarios
    e. Otra info: decisiones dejadas intencionalmente para el futuro
3. **Documentación transversal a las vistas**: como los elementos de las vistas se relacionan entre sí + justificación del uso de vistas + otra data.

Relación con Diseño

Arquitectura y diseño ambas dividen el sistema en partes, pero la principal relación o diferencia es:

- La arquitectura en si misma es una forma de diseño
- La arquitectura tiene un enfoque distinto donde se centra en la elección de componentes principales (diseño es más enfocado a la estructura interna de cada componente)
- La arquitectura es un diseño de alto nivel (se centra en decisiones de impacto significativo en la estructura general y comportamiento del sys.)
- Diseño es, como dijimos, más centrado a las vistas, debido a que se preocupa por los detalles de los componentes individuales.

Evaluación de Arquitecturas

Las arquitecturas se evalúan según **su impacto sobre atributos no funcionales** como el desempeño, confiabilidad, portabilidad, modificabilidad, etc.

Es importante tener en cuenta éstos atributos a la hora de **evaluar** una arquitectura. Ésta evaluación puede hacerse de dos formas:

1. Técnicas formales
    a. Redes de colas: herramientas matemáticas que modelan y analizan rendimiento de sistemas
    b. Model checkers: verifican que un modelo cumpla con ciertas propiedades especificadas
    c. Lenguajes de especificación: describen requisitos y propiedades del sistema de manera precisa y matemática
2. Metodologías rigurosas
    a. Metodologías de diseño arquitectónico: enfoques que ayudan a arquitectos a través de pasos y decisiones a que consideren adecuadamente los atributos no funcionales en una arquitectura.
    b. Pruebas de calidad: pruebas de carga, rendimiento, de seguridad, **análisis ATAM**, etc.
    c. Análisis de riesgo: evaluar posibles riesgos y amenazas

Análisis ATAM (se toma)

Técnica usada para evaluar y analizar una arquitectura. Pasos:

1. Recolectar escenarios: **recolecta escenarios** que describan las interacciones del sistema. Elige los de interés e incluye algunos excepcionales si se requiere.
2. Recolectar requerimientos y/o restricciones: **define lo que se espera en tales escenarios**. Lo que se espera son niveles deseados para los atributos de interés (son cosas que pueden hacer que la arqui falle). Niveles son valores.
3. Describir vistas arquitectónicas: se **recolectan las vistas** a usar
4. Análisis específicos a cada atributo: **se analizan cada uno de los atributos por separado usando cada vista recolectada**. Aquí se determinan los niveles que se pueden obtener para cada atributo y se comparan los resultados. Sirve para

decidir modificar arquitecturas. Definir en el parcial qué es un atributo: son hitos donde los escenarios nos pueden dar importancia.

5. Identificar puntos sensitivos y de compromisos: se analiza el impacto que un elemento tiene sobre **un** atributo de calidad, los de mayor impacto son **puntos de sensibilidad**. Además, se tienen **puntos de compromiso** que son elementos que son de alto impacto para **varios** atributos.

Esquema conceptual

03-arquitectura-mindmap.pdf