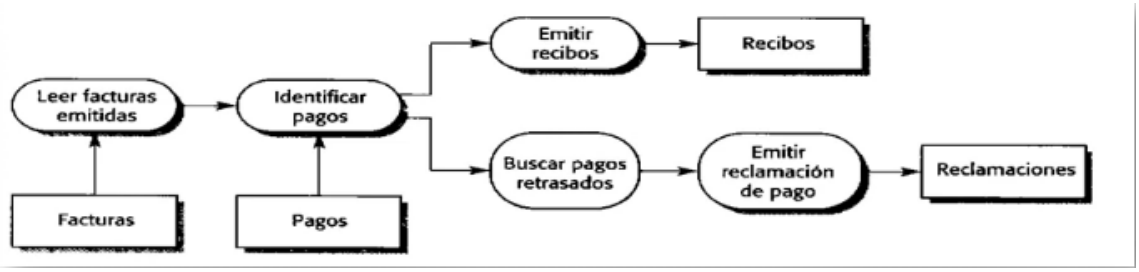
INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO

Descomposición Modular

- Una vez que organizamos el sistema, a los subsistemas los podemos dividir en módulos, se pueden aplicar los mismos criterios que vimos en la organización pero la descomposición modular es más pequeña y permite utilizar estilos alternativos.
- Estrategias de descomposición modular
  - Descomposición orientada a flujo de funciones
    - Conjunto de módulos funcionales -> se ingresan datos y se transforman en salidas.
  - Descomposición orientada a objetos
    - Conjunto de objetos que se comunican.
- Definiciones
  - Subsistema
    - Es un sistema en sí mismo cuyo funcionamiento es independiente (no depende de servicios de otros subsistemas). Los subsistemas son compuestos de módulos con interfaces definidas que se utilizan para comunicarse con otros subsistemas.
  - Módulo
    - Es un componente de un subsistema que proporciona uno o más servicios a otros módulos. A su vez utiliza servicios proporcionados por otros módulos. Por lo general no son considerados un sistema independiente.

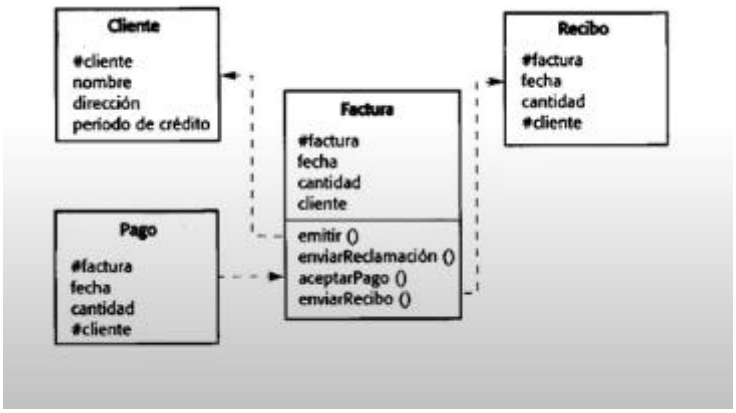
Descomposición orientada a flujo de funciones

- En un modelo de estos, los datos fluyen de una función a otra y son transformados a medida que se pasan por una secuencia de funciones hasta llegar a los datos de salida. Las transformaciones se pueden ejecutar secuencialmente o en paralelo.
- 

Descomposición orientada a objetos

- Estructura al sistema en un conjunto de objetos débilmente acoplados y con interfaces bien definidas.

INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO



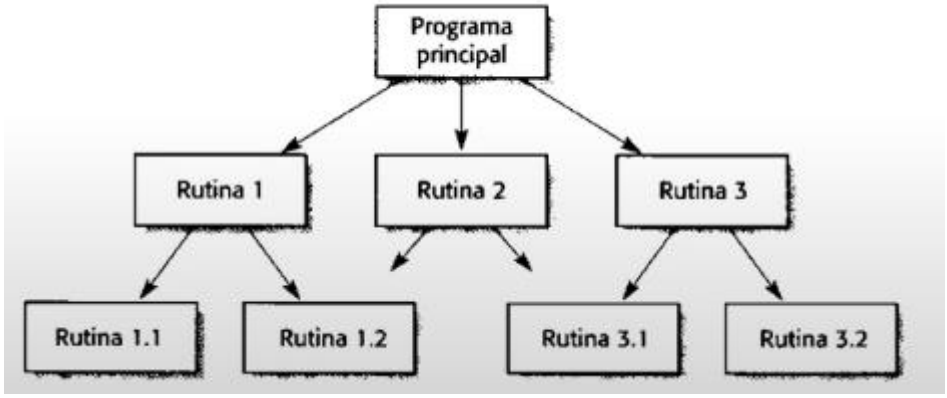
Modelos de control

- En un sistema, los subsistemas están controlados para que sus servicios se entreguen en el lugar correcto en el momento preciso.
  - Forma de ejecutar.
- Los modelos de control a nivel arquitectónico
  - Control centralizado
    - Un subsistema tiene la responsabilidad de iniciar y detener otro subsistema.
  - Control basado en eventos
    - Cada subsistema responde a eventos externos al subsistema.

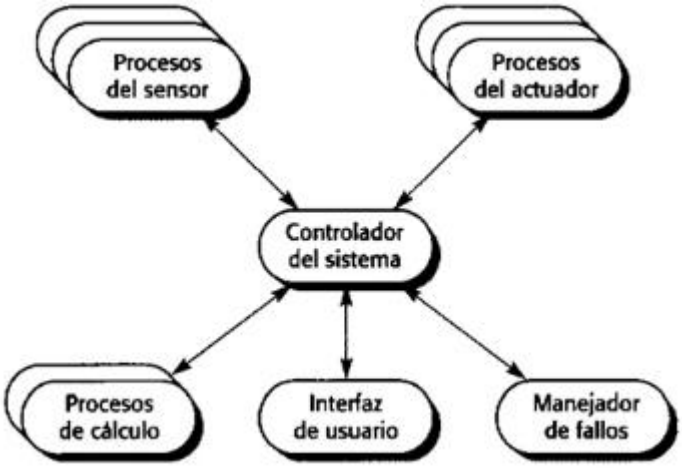
Control Centralizado

- Un subsistema es diseñado como controlador y tiene la responsabilidad de gestionar la ejecución de otros subsistemas, le ejecución puede ser secuencial o en paralelo.
  - Modelo de llamada y retorno.
    - Modelo de subrutinas que descienden.
    - Se aplica a modelos secuenciales.

INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO



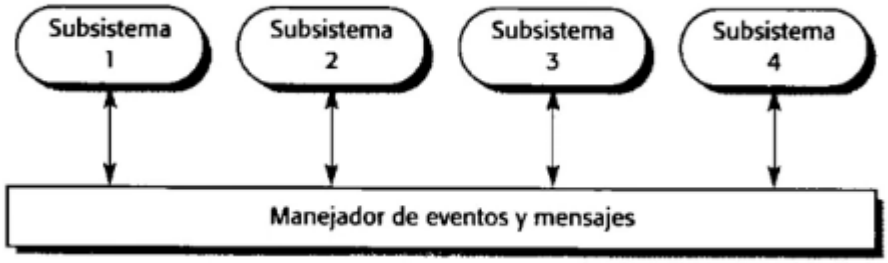
- Modelo de gestor.
  - Un gestor controla el inicio y parada coordinado con el resto de los procesos.
  - Aplicable a modelos concurrentes.



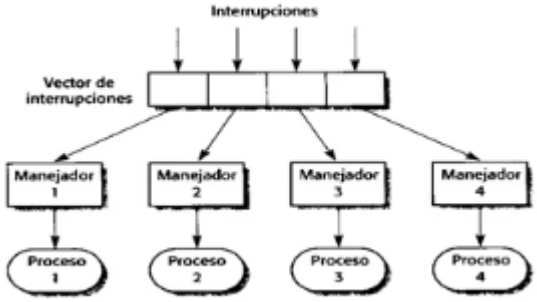
Sistemas Dirigidos por Eventos

- Se rigen por eventos generados de forma externa al proceso.
  - Eventos
    - Señal binaria.
    - Valor dentro de un rango.
    - Entrada de comando.
    - Selección del menú.
  - Modelos de sistemas dirigidos por eventos
    - Modelos de transmisión(Broadcast):
      - Un evento es transmitido a todos los subsistemas, cualquier subsistema programado para manejar ese evento lo atenderá.

INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO



- Modelos de interrupciones:
  - Se utilizan en sistemas de tiempo real donde las interrupciones externas son detectadas por un manejador de interrupciones y se envía a algún componente para su procesamiento.



Arquitectura de los Sistemas Distribuidos

- Un sistema distribuido es un sistema en el que el procesamiento de la información se distribuye sobre varias computadoras.
- Tipos genéricos de sistemas distribuidos
  - Cliente-servidor
  - Componentes distribuidos
- Características de estos sistemas
  - Compartir recursos.
  - Sistemas abiertos que no dependen de empresas en particular.
    - Diseñados con protocolos estándar.
  - Manejar aspectos de concurrencia.
    - Varios procesos en ejecución a la vez sobre varias computadoras.
  - Escalables, podemos agregar más potencia de procesamiento.
    - Añadiendo nuevos recursos que cubren nuevas demandas
  - Tolerable a fallos, si una PC cae, siguen las otras.
- Desventajas de estos sistemas

INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO

- Complejidad
  - Son más complejos que los sistemas centralizados, además del procesamiento hay que tener en cuenta los problemas de comunicación y sincronización entre equipos.
- Seguridad
  - Se accede al sistema desde varias PC generando tráfico en la red que puede ser intervenido.
- Manejabilidad
  - Las computadoras del sistema pueden ser de diferentes tipos y diferentes S.O lo que genera más dificultades para gestionar y mantener el sistema.
- Impredecibilidad
  - La respuesta depende de la carga del sistema y del estado de la red, lo que hace que el tiempo de respuesta sea variable entre peticiones.

Cuatro grandes posibilidades de arquitectura distribuida

Arquitectura Multiprocesador

- El sistema de software está formado por varios procesos que pueden o no ejecutarse en procesadores diferentes.
- La asignación de procesos a los procesadores puede ser predeterminada o mediante un dispatcher.
- Es común en sistemas grandes de tiempo real que recolecta información, toman decisiones y envían señales para modificar el entorno.

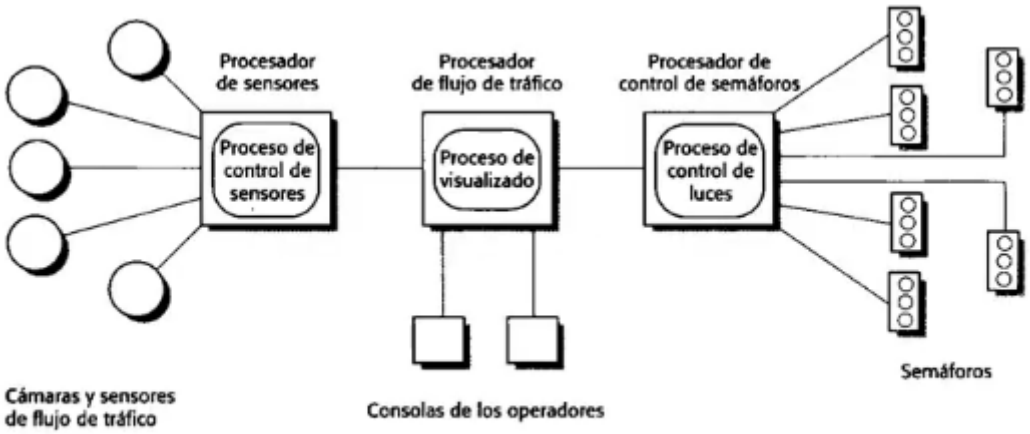


Figura 12.1 Un sistema multiprocesador de control de tráfico.

-

Arquitectura Cliente-Servidor

- Una aplicación se modela como un conjunto de servicios proporcionados por servidores y clientes que usan estos servicios.

INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO

- Cliente y servidores son procesos diferentes.
- Los servidores pueden atender varios clientes.
- Un servidor puede brindar varios servicios.
- Los clientes no se conocen entre sí.

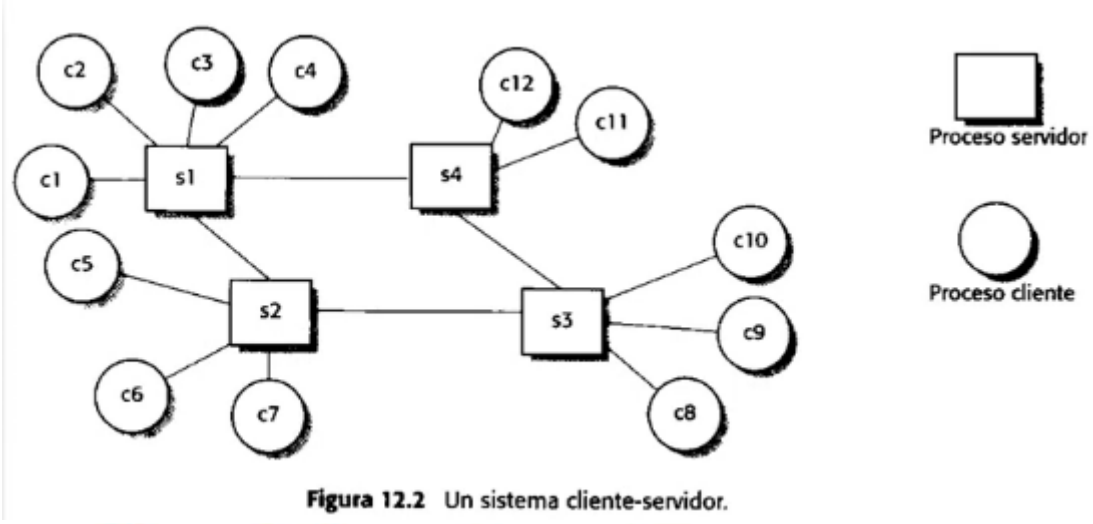
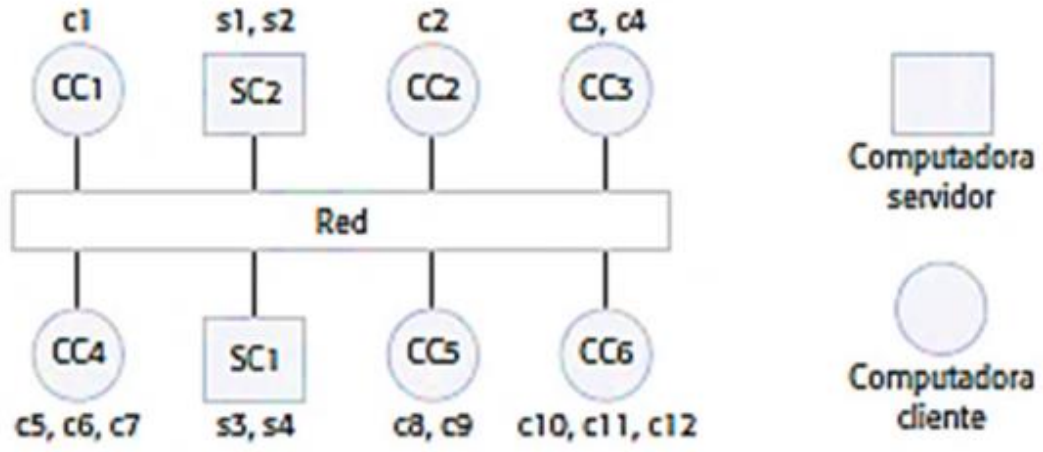
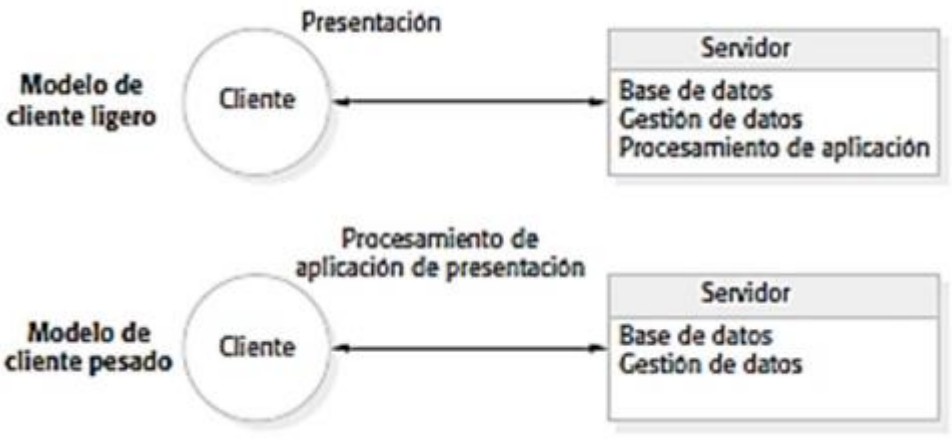


Figura 12.2 Un sistema cliente-servidor.

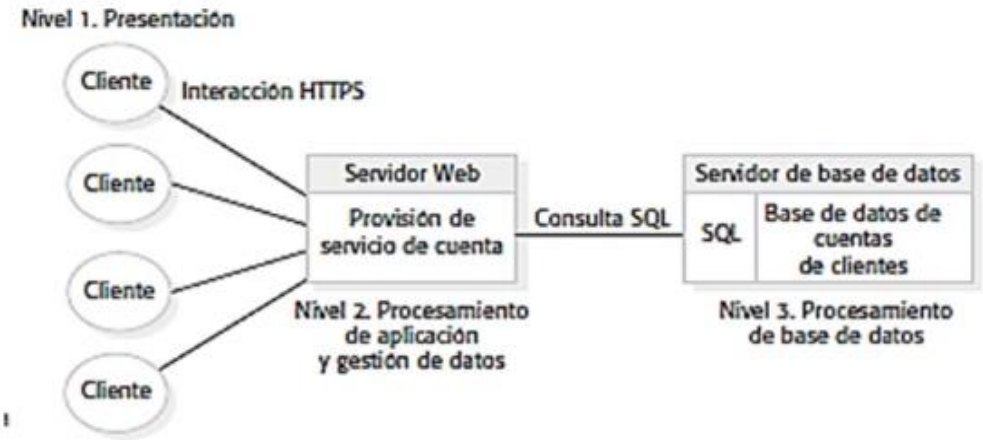


- Clasifican en niveles
  - Dos niveles:
    - Cliente ligero
      - El procesamiento y gestión de datos se lleva a cabo en el servidor.
    - Cliente pesado
      - El cliente (proceso) implementa la lógica de la aplicación y el servidor solo gestiona los datos.

INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO



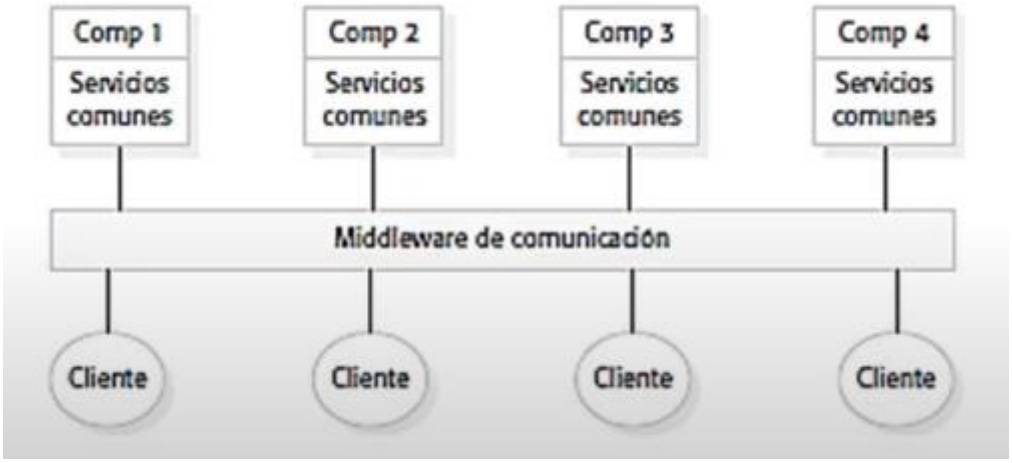
- Multinivel:
  - La presentación, procesamiento y gestión de datos son procesos lógicamente separados y se pueden ejecutar en procesadores diferentes.



Arquitectura de componentes distribuidos

- Se diseña al sistema como un conjunto de componentes u objetos que brindan una interfaz de un conjunto de servicios que ellos suministran.
- Otros componentes u objetos solicitan estos servicios.
- No hay distinción tajante entre clientes y servidores.
- Los componentes pueden distribuirse en varias máquinas a través de la red utilizando un middleware como intermediario de peticiones.

INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO

- 

Computación distribuida inter-organizacional

- Una organización tiene varios servidores y reparte su carga computacional entre ellos.
- Extender este concepto a varias organizaciones
- Arquitecturas
  - Peer-to-peer.
  - Orientados a servicios.

Arquitecturas Peer-to-Peer (P2P, Inter organizacional)

- Sistemas descentralizados en los que el cálculo puede llevarse a cabo en cualquier nodo de la red de servidores.
- Se diseñan para aprovechar la ventaja de la potencia computacional y el almacenamiento a través de una red.
- Pueden utilizar una arquitectura
  - Descentralizada
    - Donde cada nodo rutea los paquetes a sus vecinos hasta encontrar el destino.
  - Semi-centralizada
    - Donde un servidor ayuda a conectarse a los nodos o coordinar resultados.
- Ejemplos
  - Torrents, Skype, ICQ.

INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO

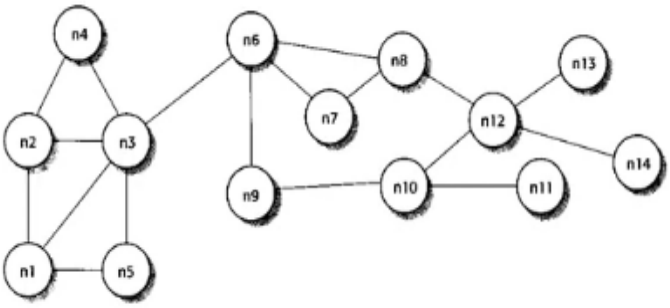
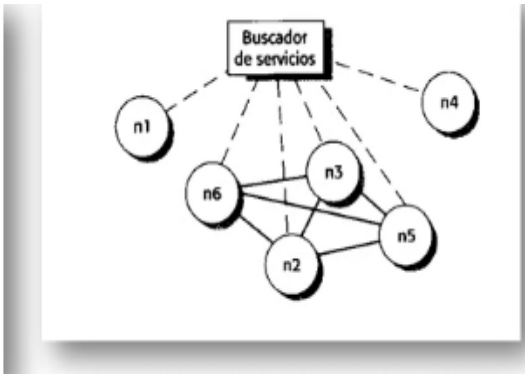


Figura 12.15 Arquitectura p2p descentralizada.



Arquitectura orientada a servicios (Inter organizacional)

- Un servicio es una representación de un recurso computacional o de información que puede ser utilizado por otros programas.
- Un servicio es independiente de la app que lo usa y puede ser construido enlazando servicios.
- Las arquitecturas de las aplicaciones de servicios web son arquitecturas débilmente acopladas.
- Funcionamiento:
  - Un proveedor de servicios que dá servicios definiendo su interfaz y su funcionalidad.
  - Para que el servicio sea externo, el proveedor publica el servicio en un "registro de servicio" con información del mismo.
  - Un solicitante enlaza este servicio a su aplicación, es decir que el solicitante incluye el código para invocarlo y procesa el resultado del mismo.

INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO

Bonus Track: Codificación

- Cuando establecemos el diseño, debemos escribir los programas que lo implementen.
- Puede ser una tarea compleja por distintos motivos:
    - Los diseñadores pueden no haber tenido en cuenta las particularidades de la plataforma y el ambiente de programación.
    - Las estructuras e interrelaciones que son fáciles de describir con diagramas , no siempre resultan ser sencillas de escribir en código.
    - Es indispensable escribir el código de forma que resulte comprensible para otras personas.
    - Se deben sacar beneficios de las características de diseño creando código que sea reusable.

Pautas Generales de Codificación

- Estas nos resultan útiles para conservar la calidad del diseño
    - Localización de entrada y salida: deseable localizarlas en componentes separados del resto del código ya que generalmente son más difíciles de testear.
    - Incluir pseudocódigo: es útil avanzar el diseño, realizando pseudocódigo para adaptar el diseño al lenguaje que usamos.
    - Revisar y reescribir, no a los remiendos: se recomienda realizar un borrador, revisarlo y reescribirlo tantas veces como sea necesario.
    - Reutilización: hay dos tipos de esta
        - Productiva: se crean componentes destinados a ser reutilizados por otra app.
        - Consumidora: se usan componentes originalmente desarrollados para otros proyectos.

Documentación

- La documentación del programa es el conjunto de descripciones escritas que explican al lector qué hace el programa y cómo lo hace.
- Se divide en:
    - Documentación interna: es concisa, escrita en un nivel apropiado para un programador. Tiene información dirigida a quienes leerán el código fuente. Incluye información de algoritmos, estructuras de control, flujos de control.
    - Documentación externa: se prepara para ser leída por quienes, tal vez, nunca verán el código real. Por ejemplo diseñadores, cuando evalúan modificaciones o mejoras.

INGENIERÍA DE SOFTWARE 2 – CLASE 8, PARTE 2: DISEÑO ARQUITECTÓNICO