FACULTAD DE INGENIERIA

Fundamentos de Ingeniería de Software

Diseño de software

Temario

- Definición
- Objetivos
- Principios de diseño
- Conceptos de diseño
- Diagramas de diseño

Objetivos (I)

- Durante el diseño se refina la arquitectura.
- El diseño es un «plano» de una solución para el sistema.
- Trata sobre **cómo** resolver un problema.

- El diseño de software es una actividad creativa donde se identifican los componentes del software y sus relaciones, con base en los requerimientos de un cliente.

Objetivos (2)

- El diseño de un sistema es ***correcto*** si un sistema construido de acuerdo a ese diseño satisface los requerimientos del sistema
- Pero el objetivo del diseño no es encontrar **el** diseño correcto, sino:
  - encontrar el mejor diseño posible
  - dentro de las limitaciones impuestas por
    - los requisitos,
    - el ambiente físico del sistema
    - y el ambiente social donde el mismo va a operar
    - ¿otras limitaciones?

Diseño (I)

- Un diseño claramente debe ser:
  - Verificable
  - Completo (implementa toda la especificación)
  - Rastreable. Se puede rastrear hacia los requisitos que diseña (*traceable*)
- Sin embargo, las dos cosas más importantes concernientes a los diseñadores es que el diseño sea:
  - Eficiente
  - Simple
  - *¿Por qué?*

Diseño (2)

- Crear un diseño simple y eficiente de un sistema «grande» puede ser una tarea extremadamente compleja.
  - Actividad básicamente creativa
  - No puede reducirse a una serie de pasos a seguir
  - Sin embargo, se pueden dar guías
- Si se logra manejar la complejidad,
  - se reducen los costos del diseño
  - se reduce la posibilidad de introducir defectos durante el diseño

Principios de diseño

- Dividir y conquistar
- Abstracción
- Modularidad
- Reuso

- Tres conceptos claves para la calidad de un diseño
  - Acoplamiento (bajo)
  - Cohesión (alta)
  - Principio abierto-cerrado (cumplir con el principio)

Dividir y conquistar

- Debido a la complejidad de los grandes problemas y las limitaciones de la mente humana estos no se pueden atacar como una unidad monolítica.
  - Aplicar dividir y conquistar
  - Dividir en piezas que pueden ser «conquistadas» por separado. Sino se hizo una división poco inteligente
- Dividir en piezas con esta propiedad es una tarea compleja para el diseño de software.

Abstracción

- La abstracción es «una vista de un objeto que se enfoca en la información relevante para un propósito particular e ignora el resto de la información».
- Permite al diseñador considerar una componente a un cierto nivel de abstracción sin preocuparse por los detalles de implementación de dicha componente.
- Se describe el comportamiento exterior sin describir los detalles internos.

Modularidad

- Un sistema se considera modular si
  - el sistema consiste de componentes
  - y estos se pueden implementar separadamente
  - y el cambio en una tiene mínimos impactos en otras componentes.
- El objetivo es colocar diferentes funcionalidades y responsabilidades en diferentes componentes.

Reuso

- Utilizar nuevamente algo (o construir pensado en)
  - Costos de pensar en reuso a futuro, costo de reusar algo.
- Hay varios niveles
  - Objeto, componente, sistema
- Beneficios
  - Mayor confianza: ya probado en producción
  - Reducción del riesgo: costo conocido
  - Conformidad con estándares
  - Desarrollo acelerado: mejora velocidad y *time to market*
- ¿Qué problemas pueden ocurrir?

Acoplamiento

- Captura la *fuerza* de interdependencia entre módulos.
- Cuánto más acoplados, más dependientes entre sí.
  - Entonces, más difícil comprenderlos y modificarlos
- El grado de acoplamiento entre módulos depende de:
  - cuánta información se necesita sobre el otro módulo para entenderlo (la menor posible)
  - qué tan compleja (simple)
  - qué tan explícita es esta información (visible o identificable)

Cohesión

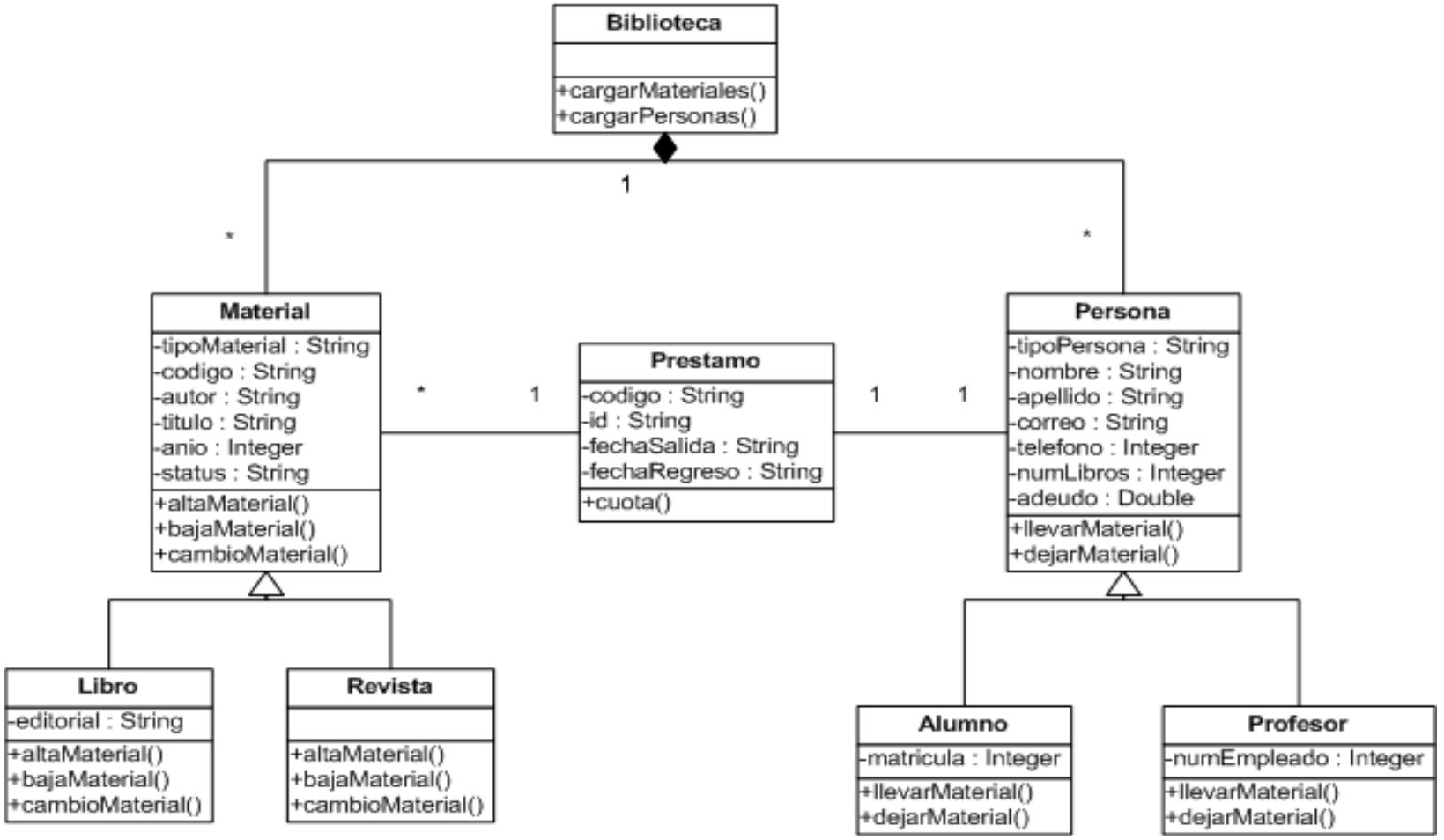
- Se focaliza en conocer porqué los elementos de un módulo están juntos en ese módulo.
- El objetivo es tener en un mismo módulo elementos que están **fuertemente vinculados.**
  - Módulos más fáciles de entender y modificar.

Principio abierto-cerrado

- Objetivo (otra vez): promover la construcción de sistemas que sean fáciles de modificar.
- Módulos **abiertos** para la **extensión**
  - El comportamiento puede ser extendido.
- Módulos **cerrados** para la **modificación**
  - Extremo: cuando se hacen mejoras, no es necesario tocar el código existente.
  - La interfaz no debe cambiar y se debe asegurar que se mantiene el comportamiento.

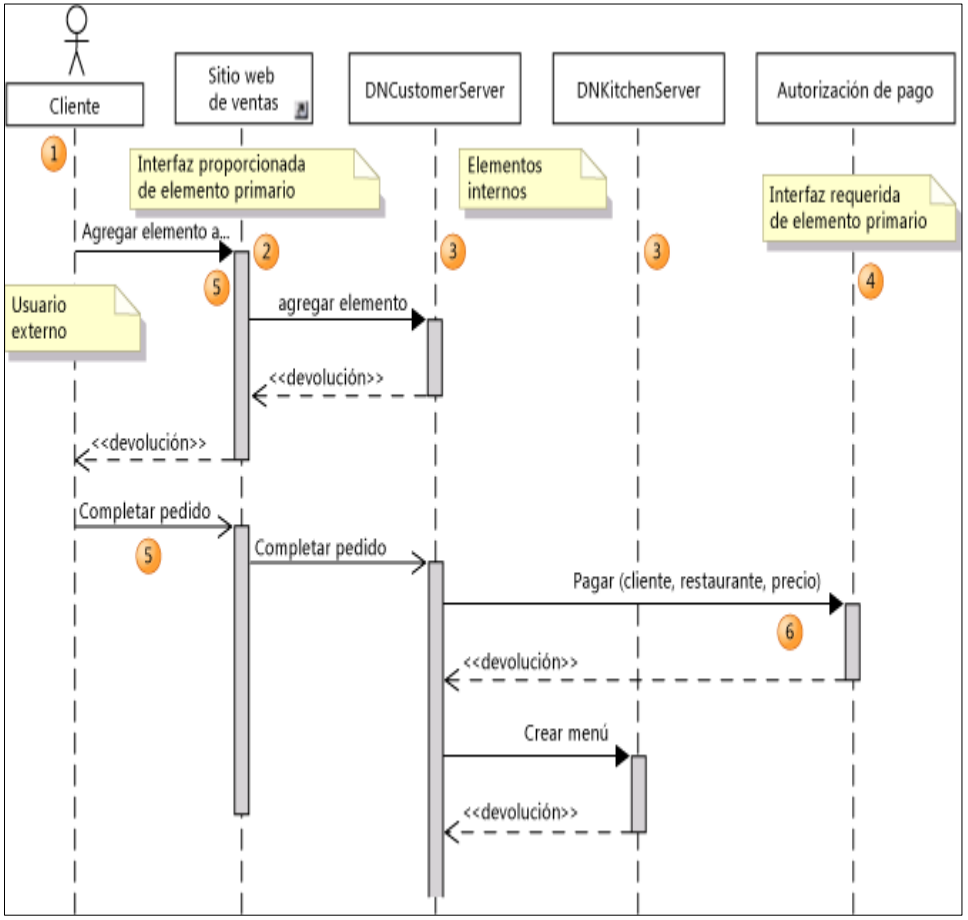
UML – Diagramas de diseño (I)

- Diagrama de clases



UML – Diagramas de diseño (2)

- Diagramas de secuencia y colaboración



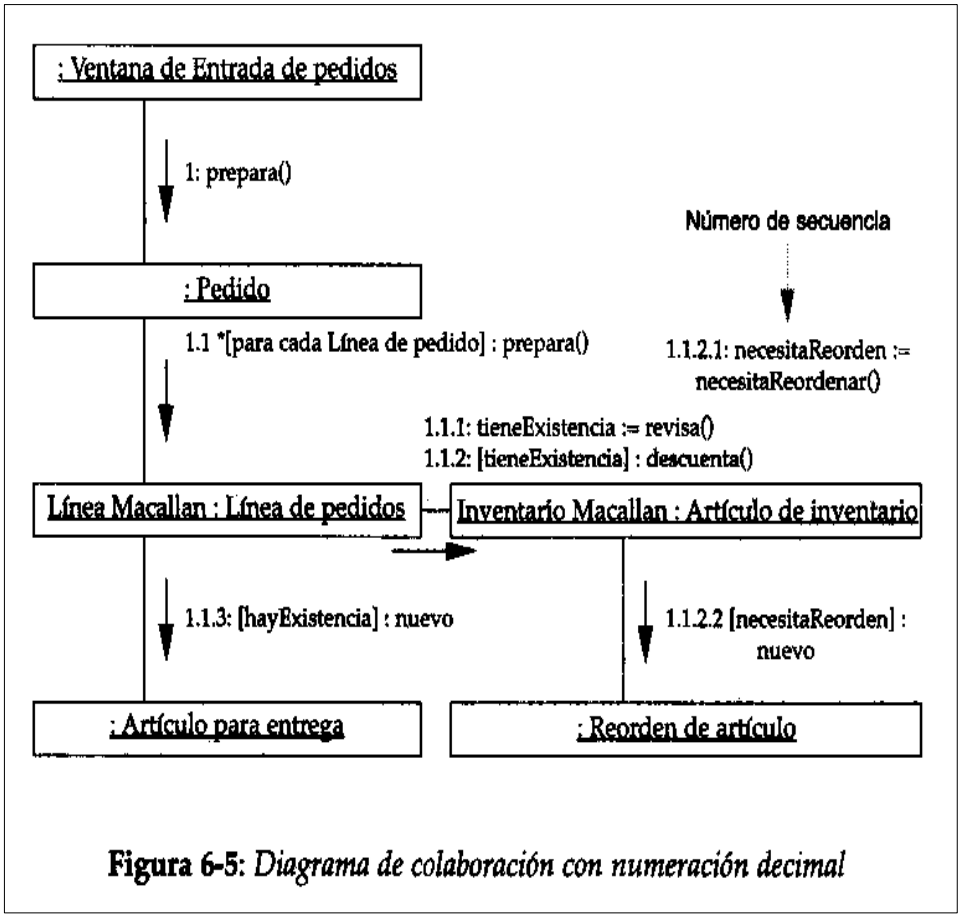
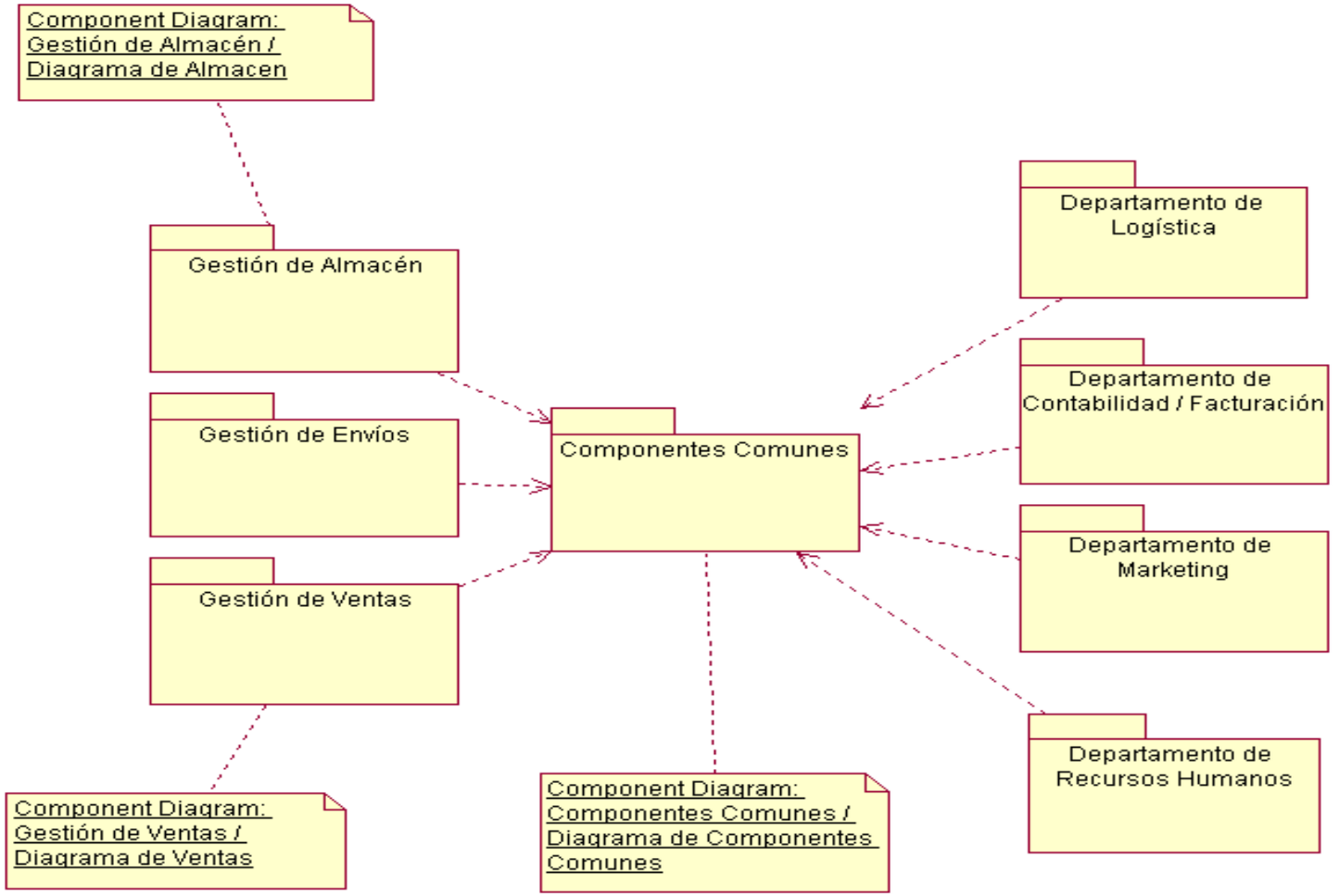


Figura 6-5: *Diagrama de colaboración con numeración decimal*

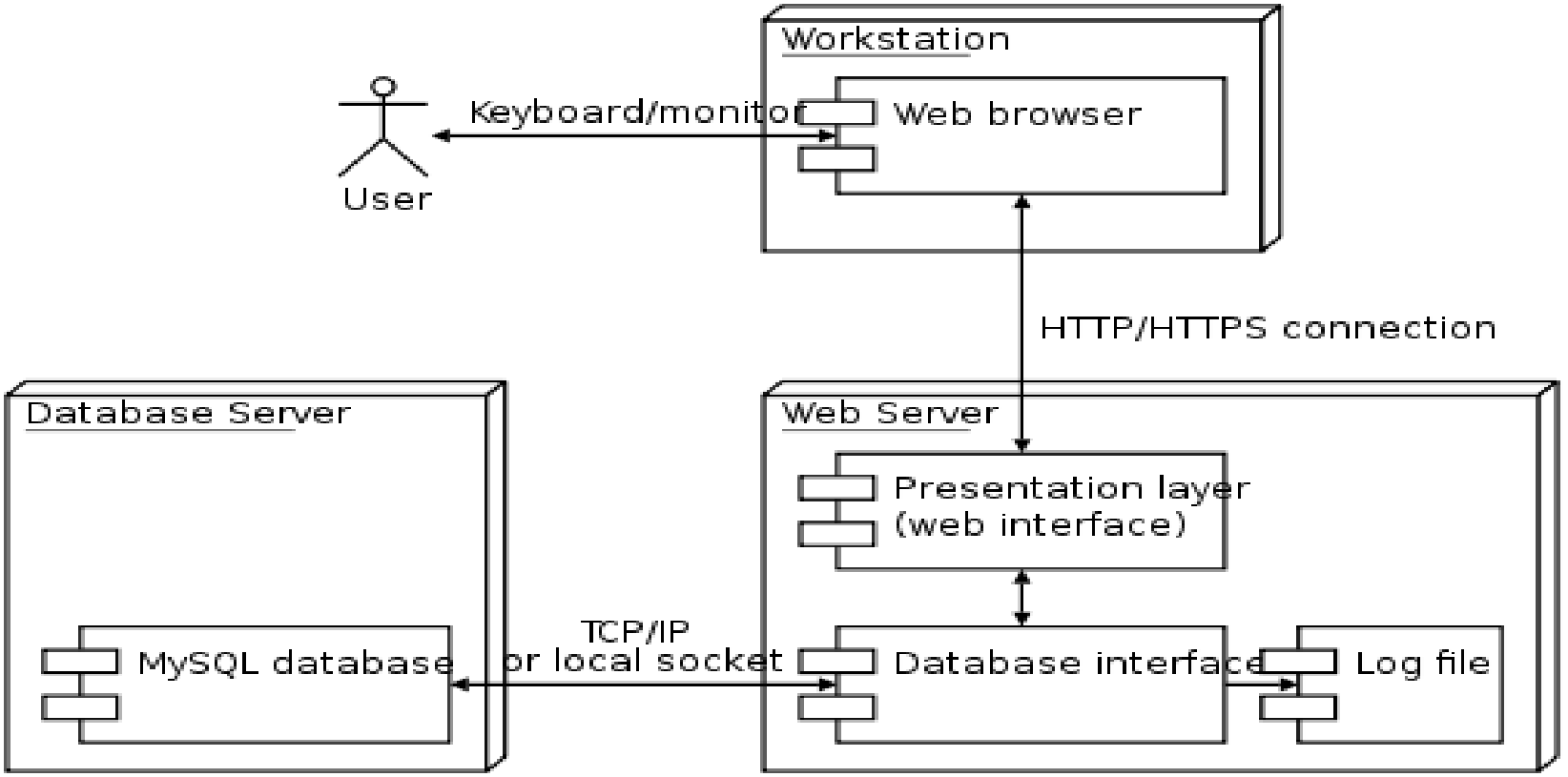
UML – Diagramas de diseño (3)

- Diagramas de paquetes



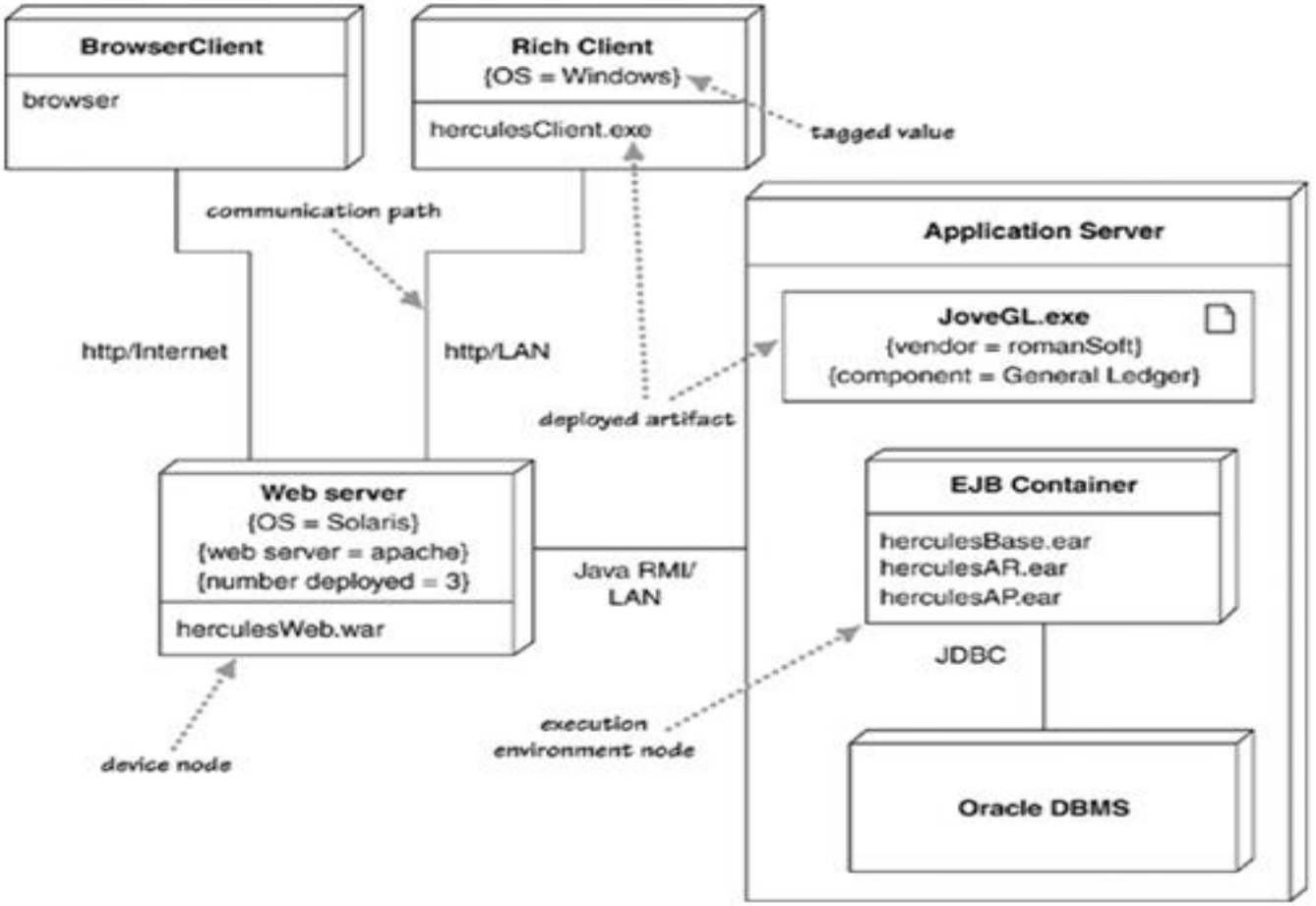
UML – Diagramas de diseño (4)

- Diagramas de componentes



UML – Diagramas de diseño (5)

- Diagramas de despliegue



Próxima clase:

Pruebas de software