Fundamentos de la programación con componentes (Parte II)

Luis G. Montané Jiménez
Tópicos Selectos de Computación I
Facultad de Estadística e Informática
Agosto de 2014

Agenda

- Introducción
- Componentes
- **Arquitectura de Software**
- Estructura de un componente

¿Qué es la arquitectura de un sistema?

- La vista general sobre la composición del sistema
  - Componentes y relaciones
  - Distribución
  - Estructura
  - Heterogeneidad: ligas con sistemas externos, diferentes plataformas, etc

¿Por qué una arquitectura es importante?

Muchos sistemas terminan por ser un desastre

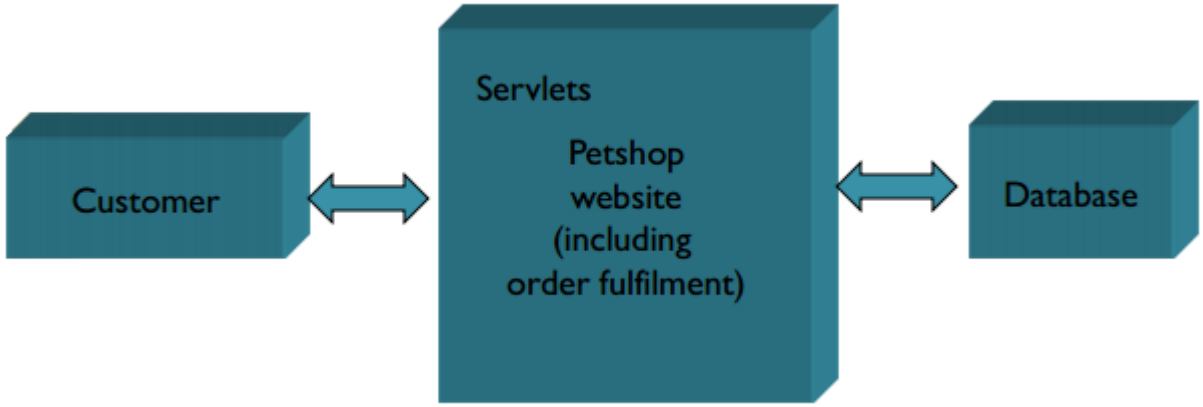
- Limitado rol dado a la arquitectura
- Arquitectura se convierte en un factor que se considera tardíamente en el desarrollo de software

¿Cómo el diseño de una arquitectura beneficia el desarrollo de software?

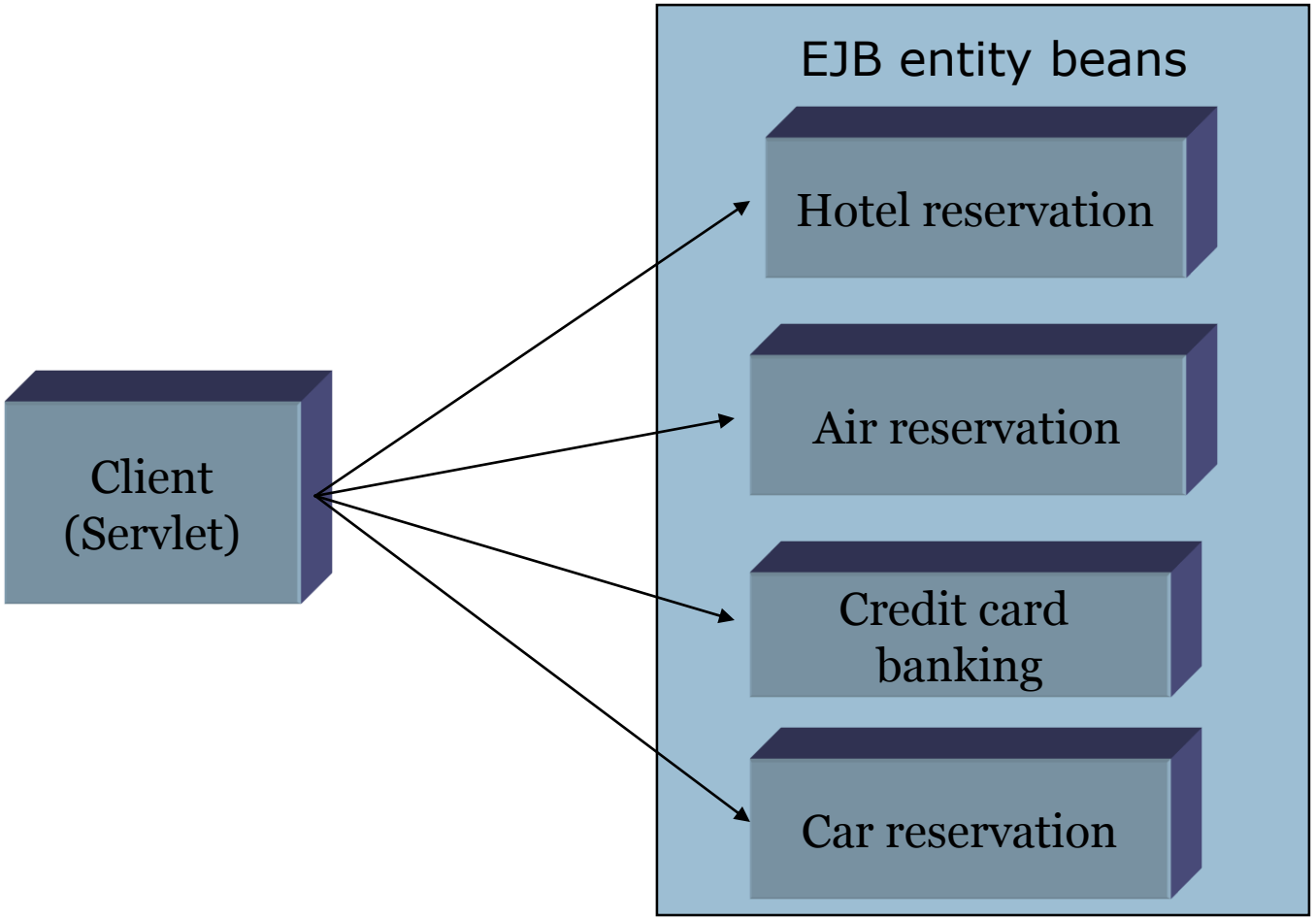
- El manejo de proyectos: en el desarrollo top-down, componentes pueden ser asignados a diferentes grupos de trabajo
- La coordinación centralizada de proyectos: se conoce como todas las piezas deben ser colocadas
- Las decisiones acerca del middleware, plataforma, estructura deben ser centralizadas

Mala arquitectura = escalabilidad pobre y difícil mantenimiento

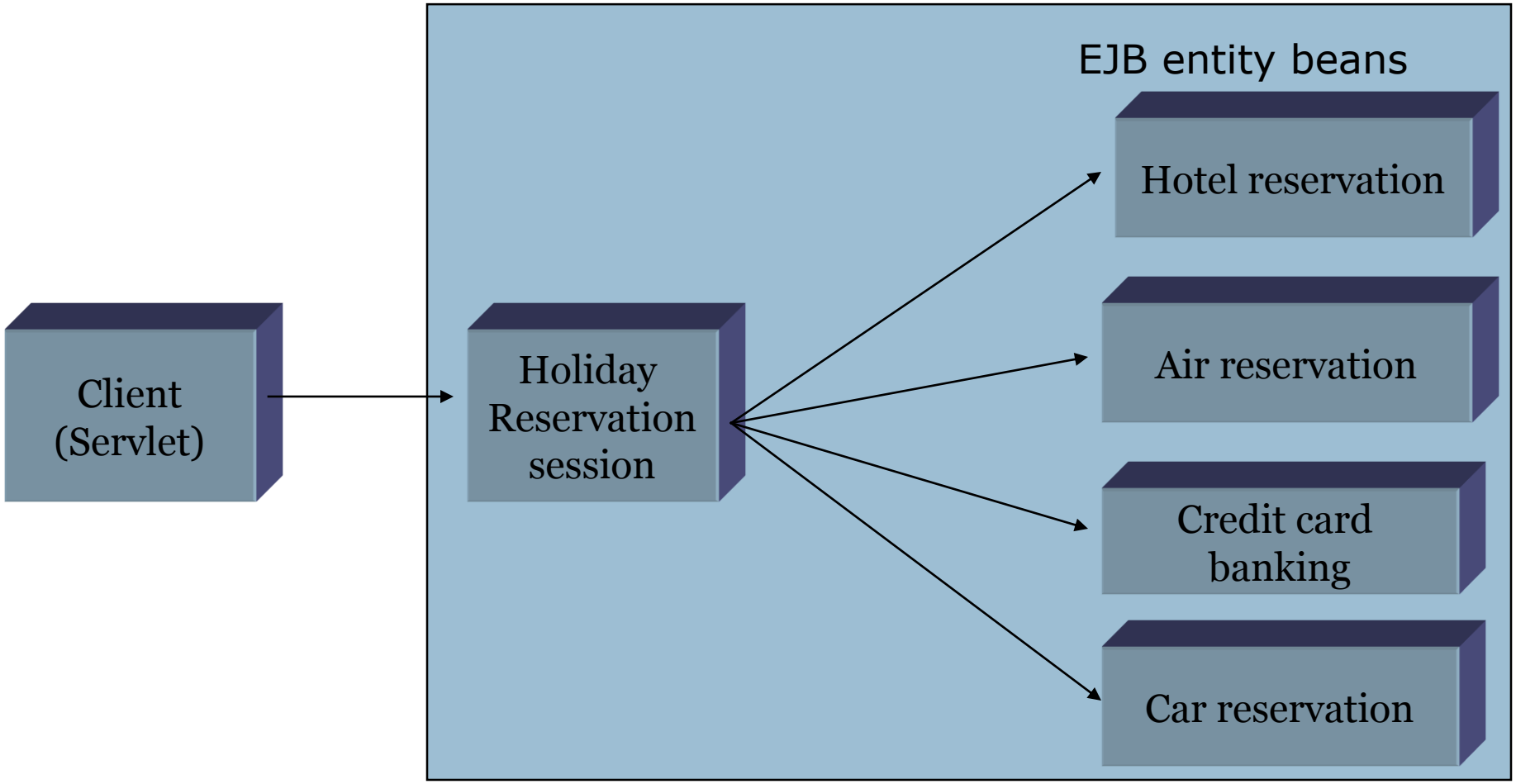
- Toda la lógica del negocio esta codificada en el website
- Difícil de mantener, ya que todo esta mezclado



Mala arquitectura: mal desempeño



Buena arquitectura = buen desempeño



Definiciones de arquitectura de software

- Perry & Wolf
  - Arquitectura de software = {Elementos (Qué), Forma (Cómo), Razonamiento (Por qué)}
- Kruchten
  - Trata del diseño e implementación del más alto nivel de estructura del software
  - Arquitectura = abstracción, descomposición, estilo, estética
- Shaw & Garlan
  - Arquitectura de software [es un nivel de diseño que] comprende la descripción
    - Elementos de cuales el sistema está construido
    - Interaccionen entre estos elementos
    - Patrones que guían esta composición y restricciones de estos patrones

Agenda

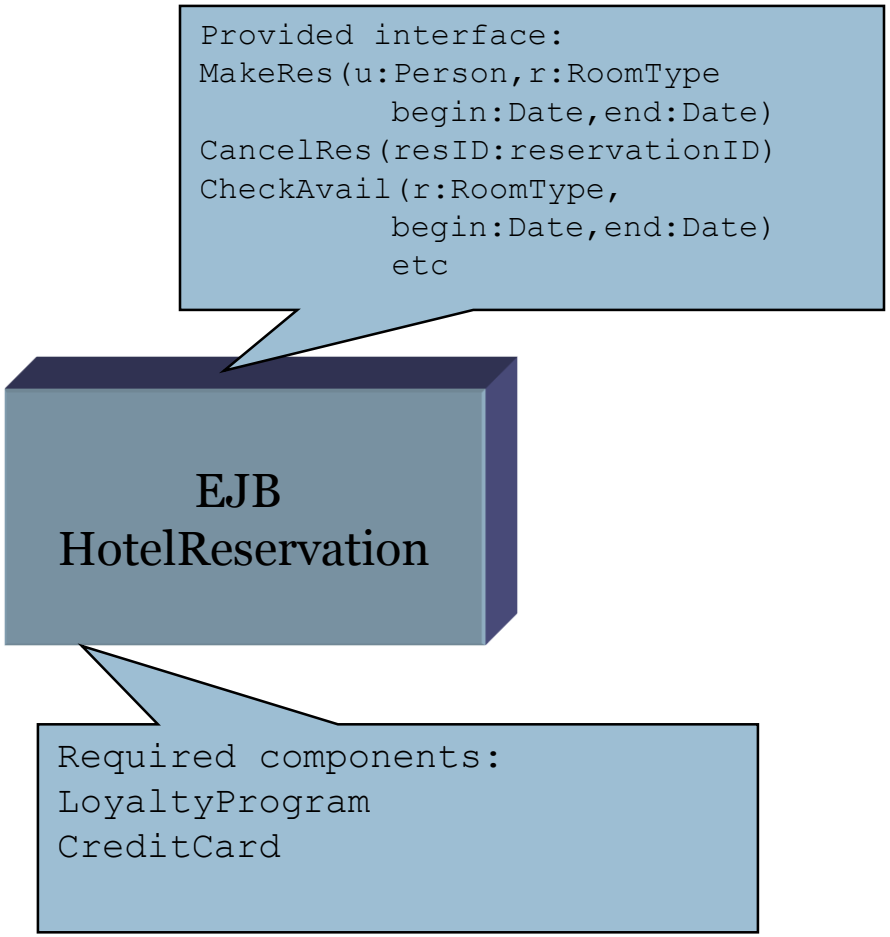
- Introducción
- Componentes
- Arquitectura de Software
- **Estructura de un componente**

Conceptos clave

- Existen tres bloques de construcción en la descripción de una arquitectura de software:
  - Componentes
  - Conectores
  - Configuración

Ejemplo: Componente

- Este componente para la reservación de hotel está programado como un EJB
- Se comunica con los clientes vía una interfaz
- Es a su vez un cliente que utiliza otros componentes



Conectores

- Un ***conector*** es un elemento arquitectural que modela
  - Interacciones de uso entre componentes
  - Reglas que gobiernan estas relaciones
- Simple interacciones
  - Llamadas a procedimientos
  - Accesos a variables compartidas
- Complejas y semánticamente interacciones
  - Protocolos Cliente-Server
  - Protocolos de acceso a bases de datos
  - Eventos asíncronos
- Nuestra definición:
  - *Un elemento de un sistema utilizado para definir la comunicación entre componentes*

¿Qué es un componente?

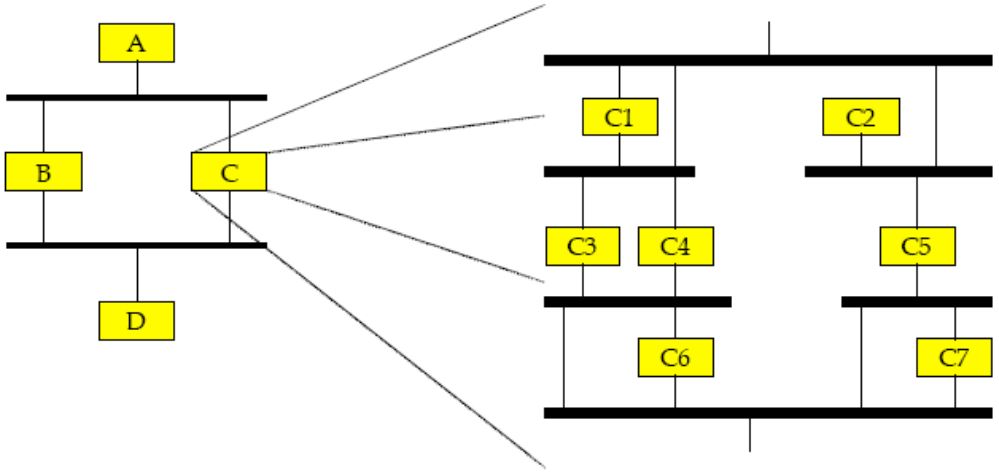
- Un componente es:
  - Paquete de software (modular, encapsula un conjunto de funciones y es reutilizable).
  - Puede ser independientemente remplazado (substituible) .
  - Provee y requiere servicios ambos basados en interfaces especificas.

Interfaces

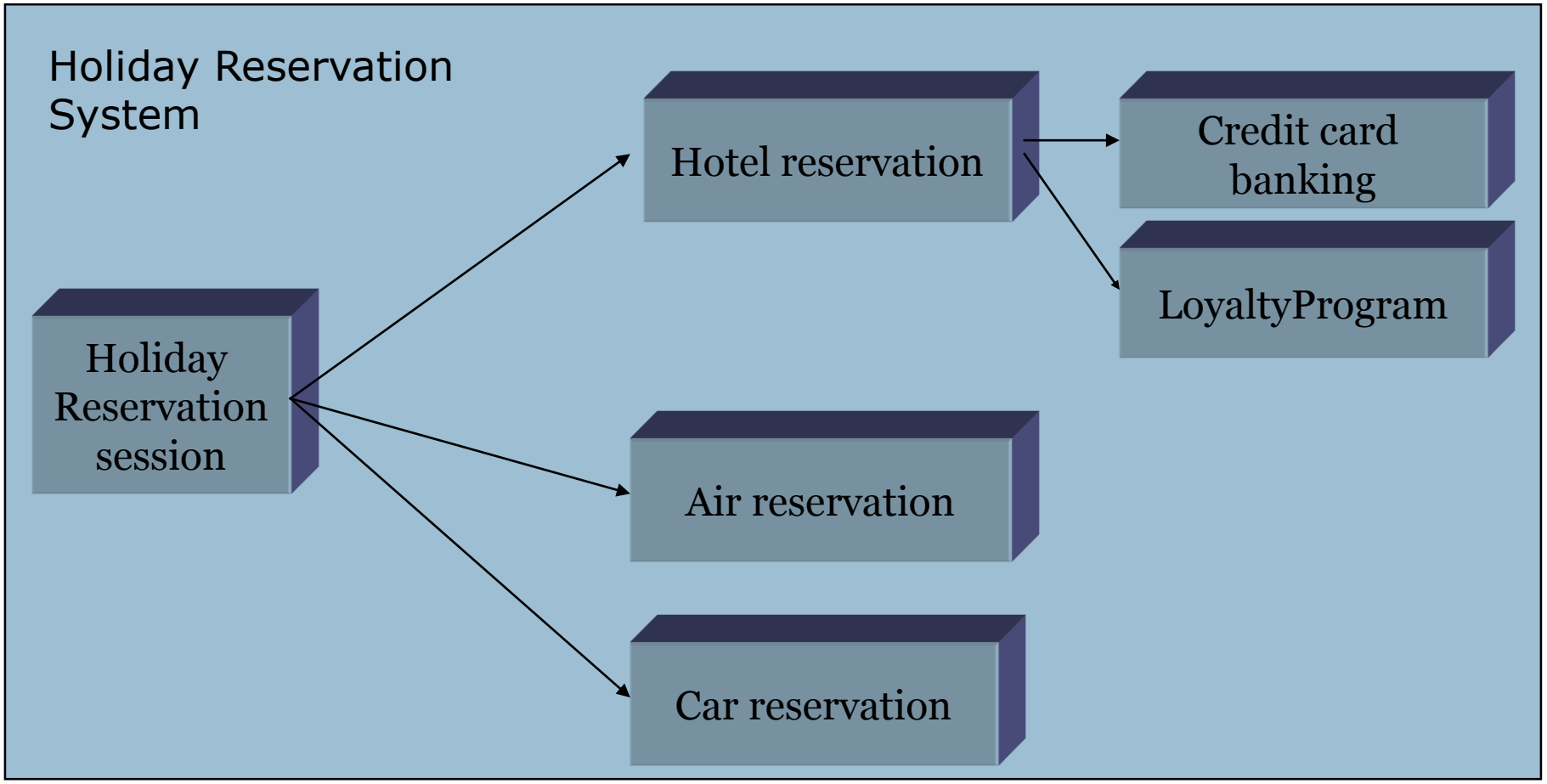
- Las interfaces son colecciones de uno o más métodos que pueden o no contener atributos.

Configuración o topologías

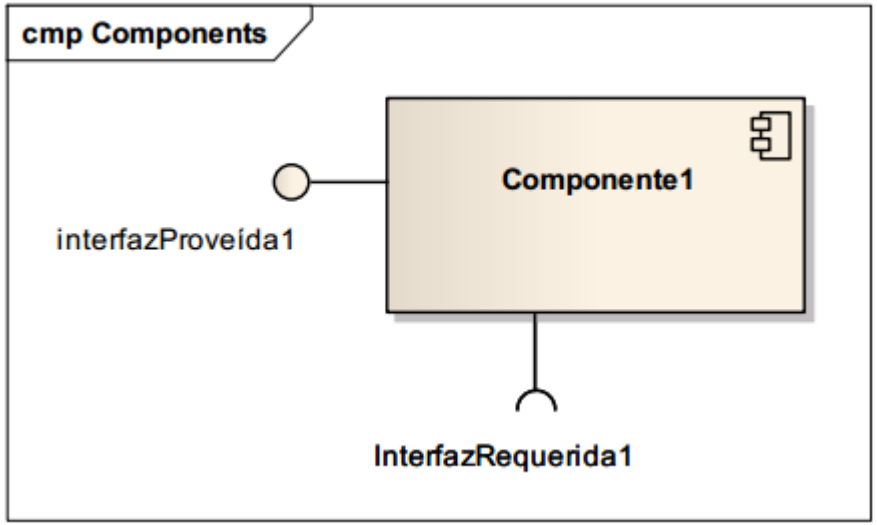
- Una configuración o topología es un grafo que une componentes con conectores que describe la estructura de una arquitectura



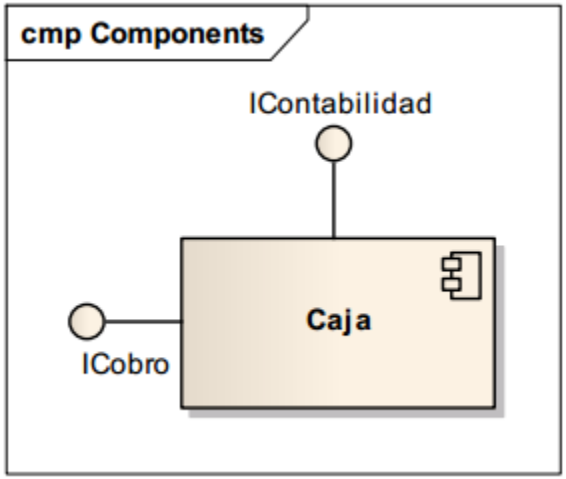
Configuración



Representación

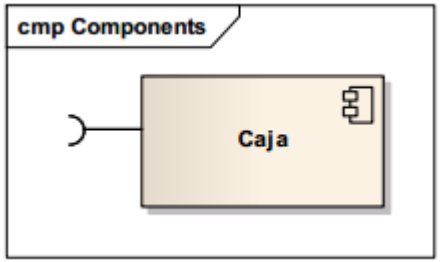


Componente que provee dos interfaces



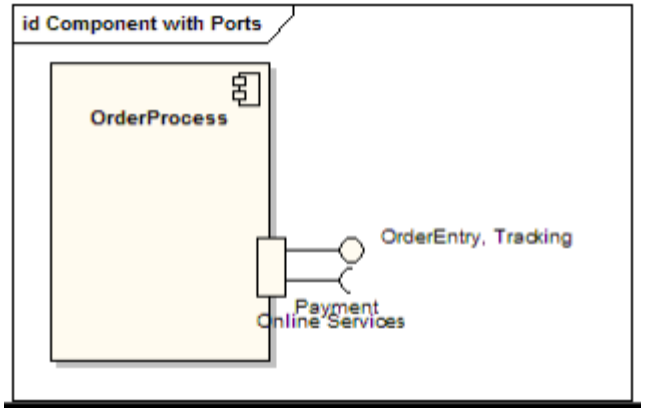
Interfaz requerida

- El conector Ensamble une la interfaz requerida del componente (Componente1) con la interfaz proporcionada de otro componente (Component2); esto permite que un componente provea los servicios que otro componente requiere.



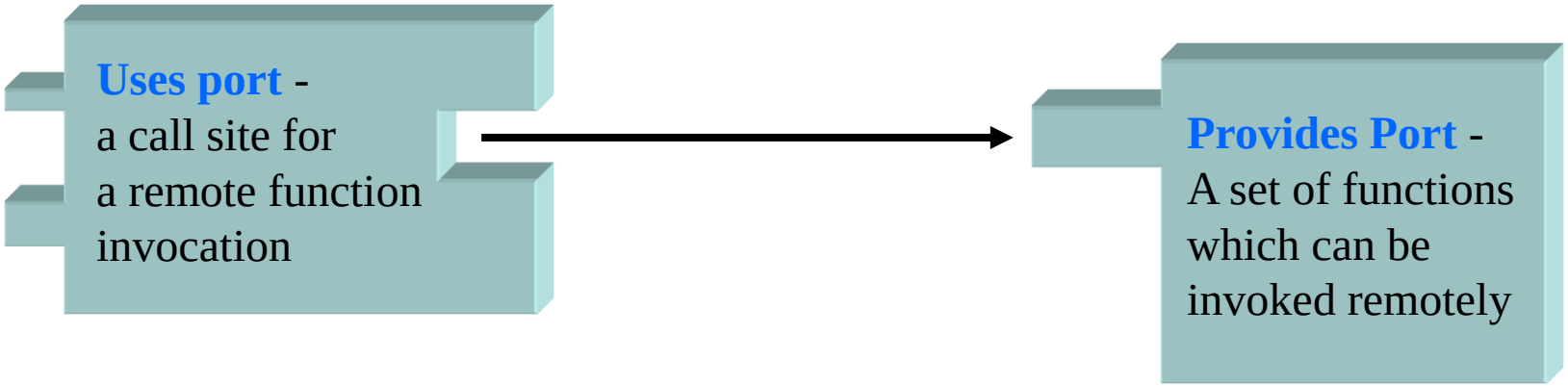
Puertos

- Usar puertos permite que se especifique un servicio o comportamiento a su entorno así como también un servicio o comportamiento que un componente requiere. Los puertos pueden especificar entradas, salidas así como también operar bidireccionalmente.

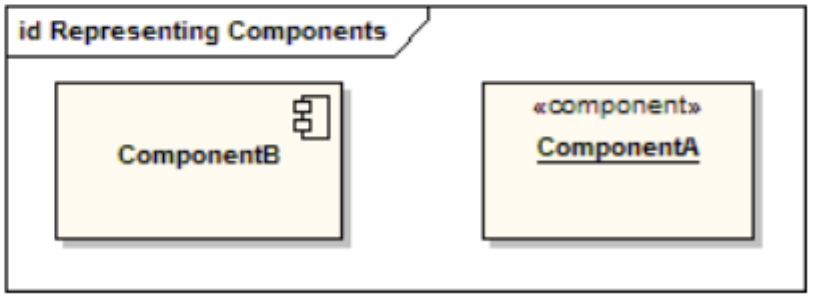


Puertos

- Los componentes pueden ser conectados a través de sus interfaces (ports) a través de los cuales un componente invoca los servicios de otro
- Existen dos tipos de interfaces (ports)
  - La interfaz que provee (provides port): implementa una interfaz remota
  - La interfaz que requiere (uses port): usa una interfaz remota
  - Un componente que usa y otro que provee pueden ser conectados



Representación



Clasificación de diagramas

- Estáticos
  - Diagramas de casos de uso
  - Diagramas de clases
- Dinámicos
  - Diagramas de estado
  - Diagramas de actividad
  - Diagramas de secuencia
  - Diagramas de colaboración
- Implementación
  - ***Diagrama de componentes***
  - ***Diagrama de despliegue***

Diagrama de clases

- Es un diagrama estático que describe la estructura de un sistema. Muestra las clases, atributos y las relaciones entre ellos.
- Los diagramas de clases se utilizan durante el análisis y diseño de sistemas.
- Presenta el diseño conceptual de la información que se manejará en el sistema.

Diagrama de clases

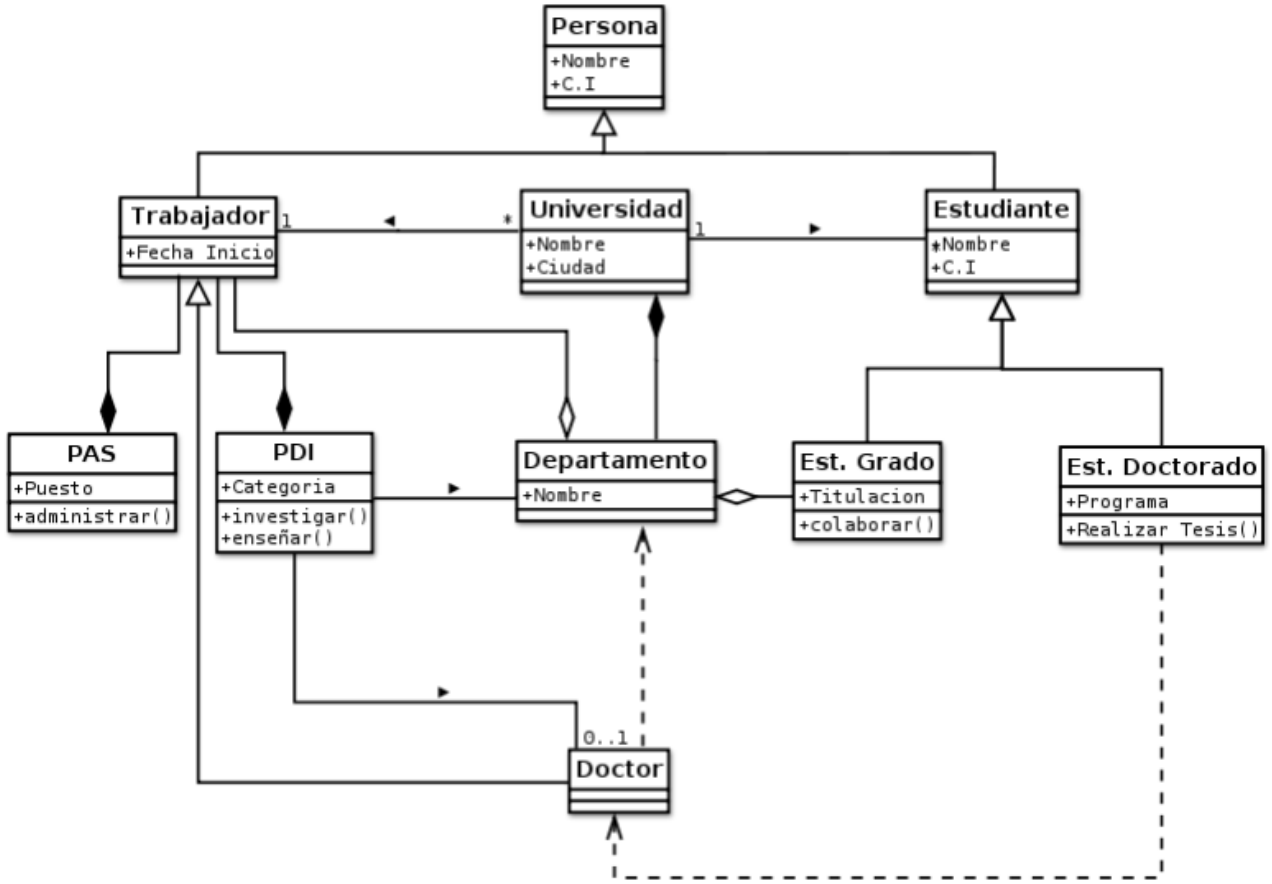


Diagrama de clases

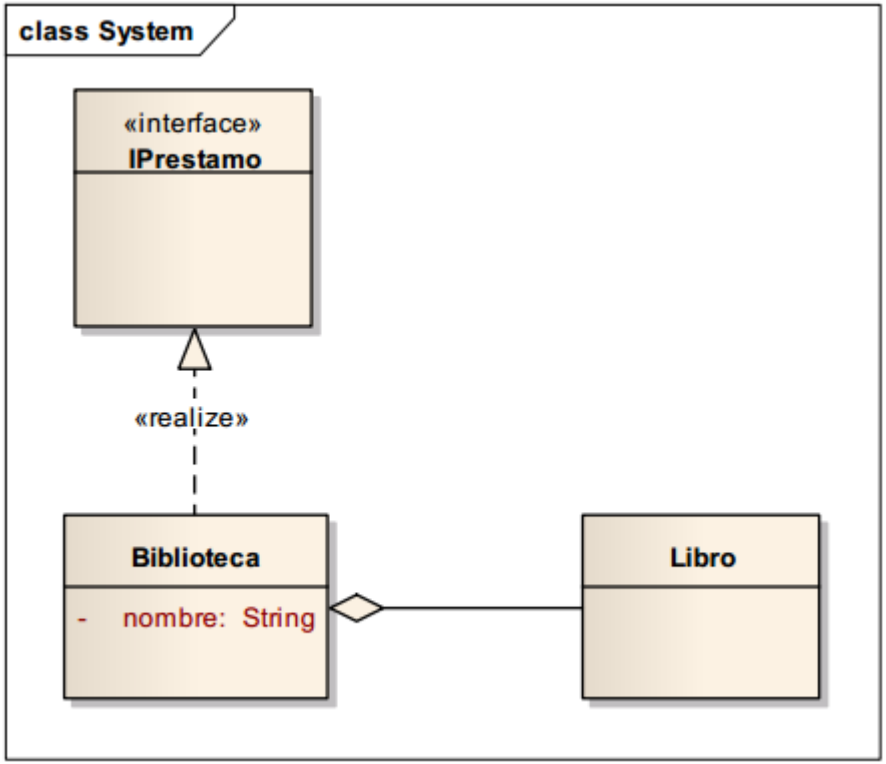
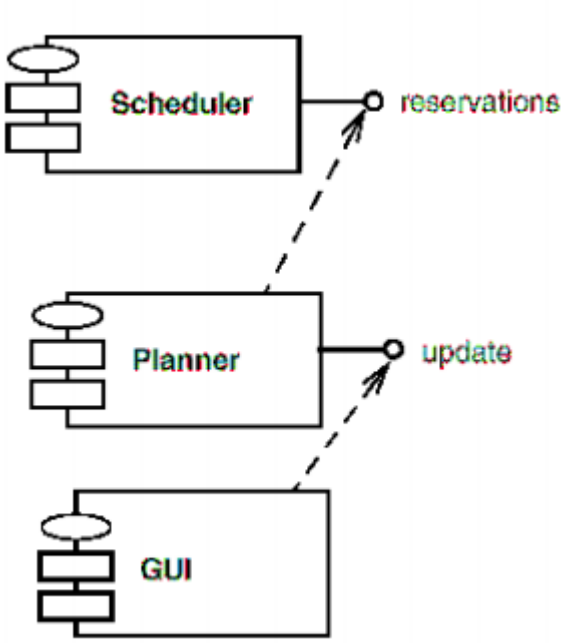
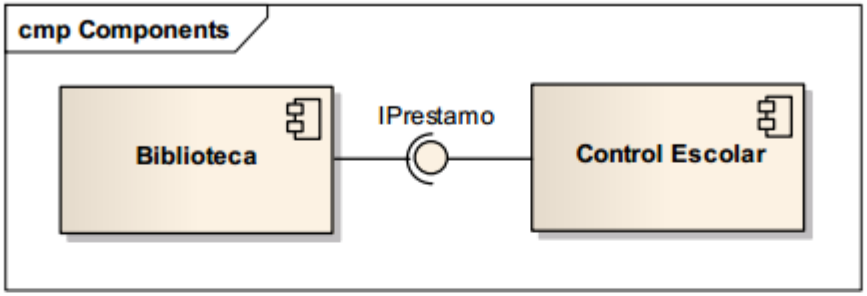


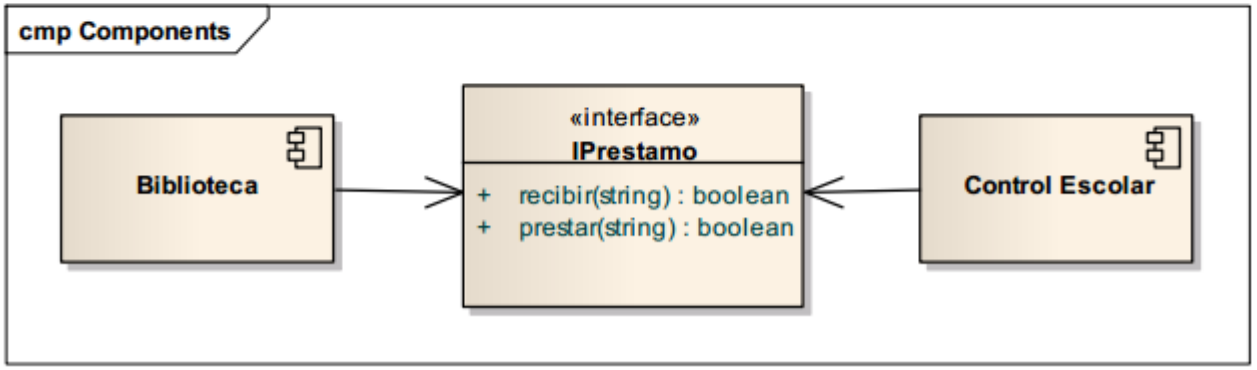
Diagrama de componentes

- Un diagrama de componentes muestra la organización y dependencias entre un conjunto de componentes.

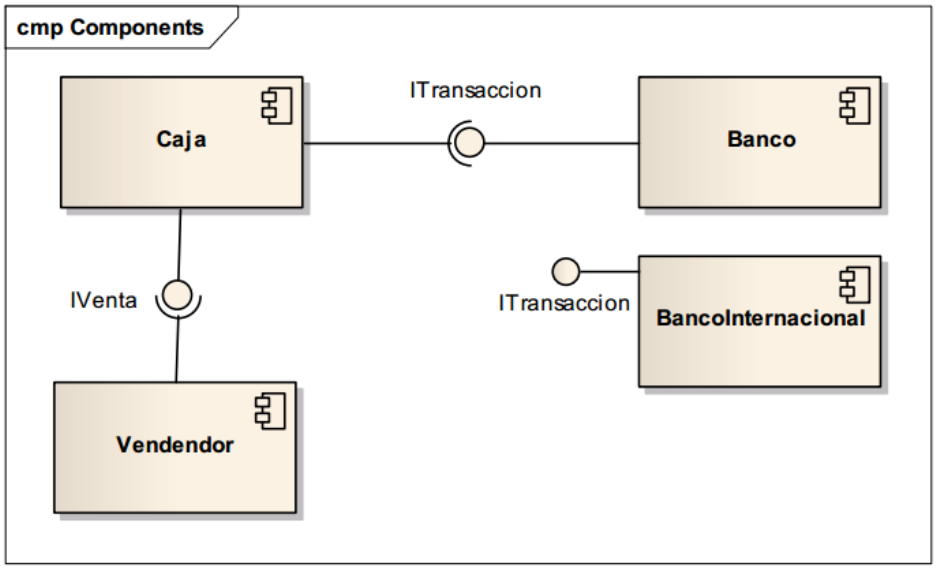


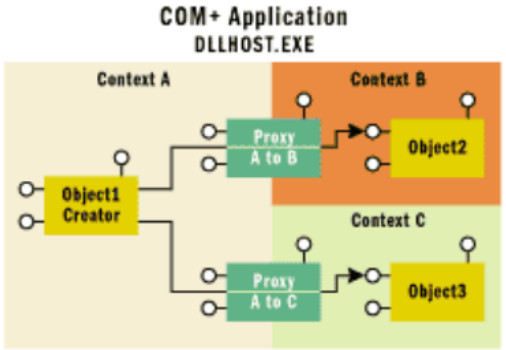
Representación





Representación





Ejemplos

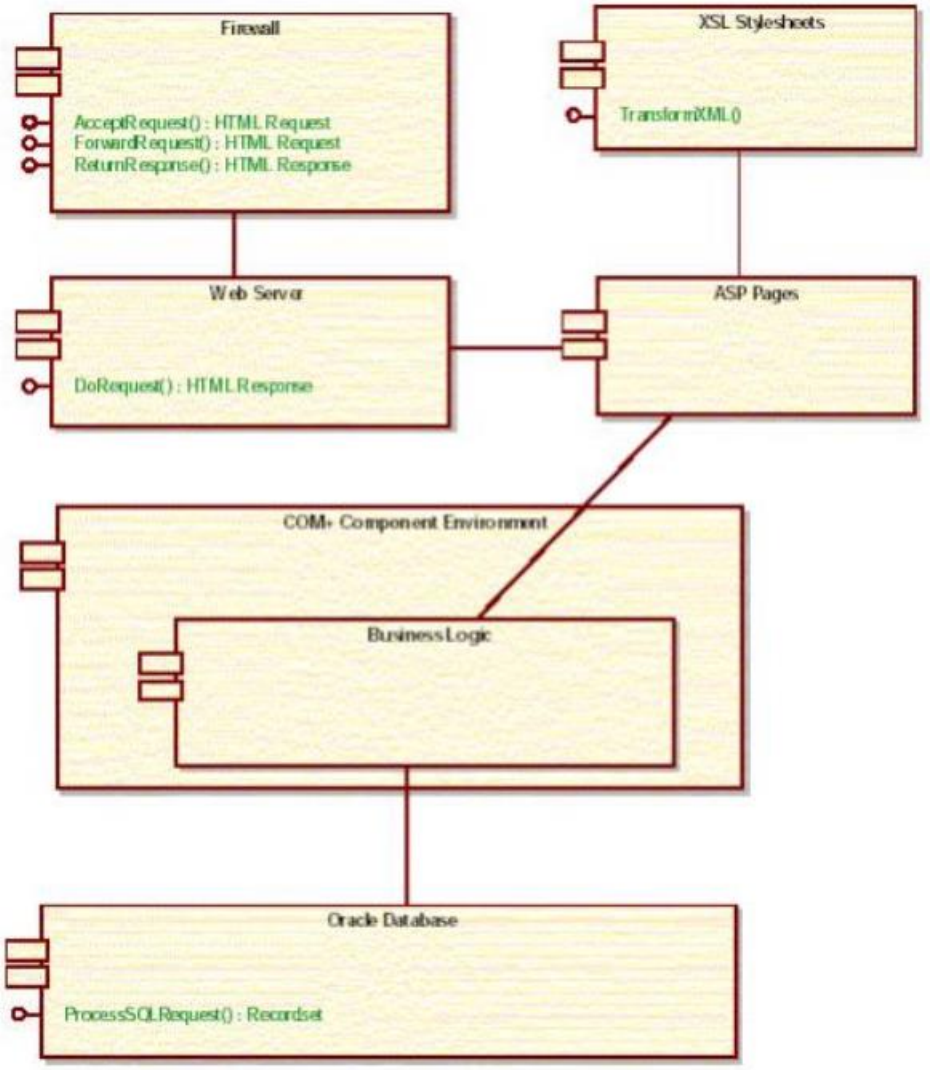
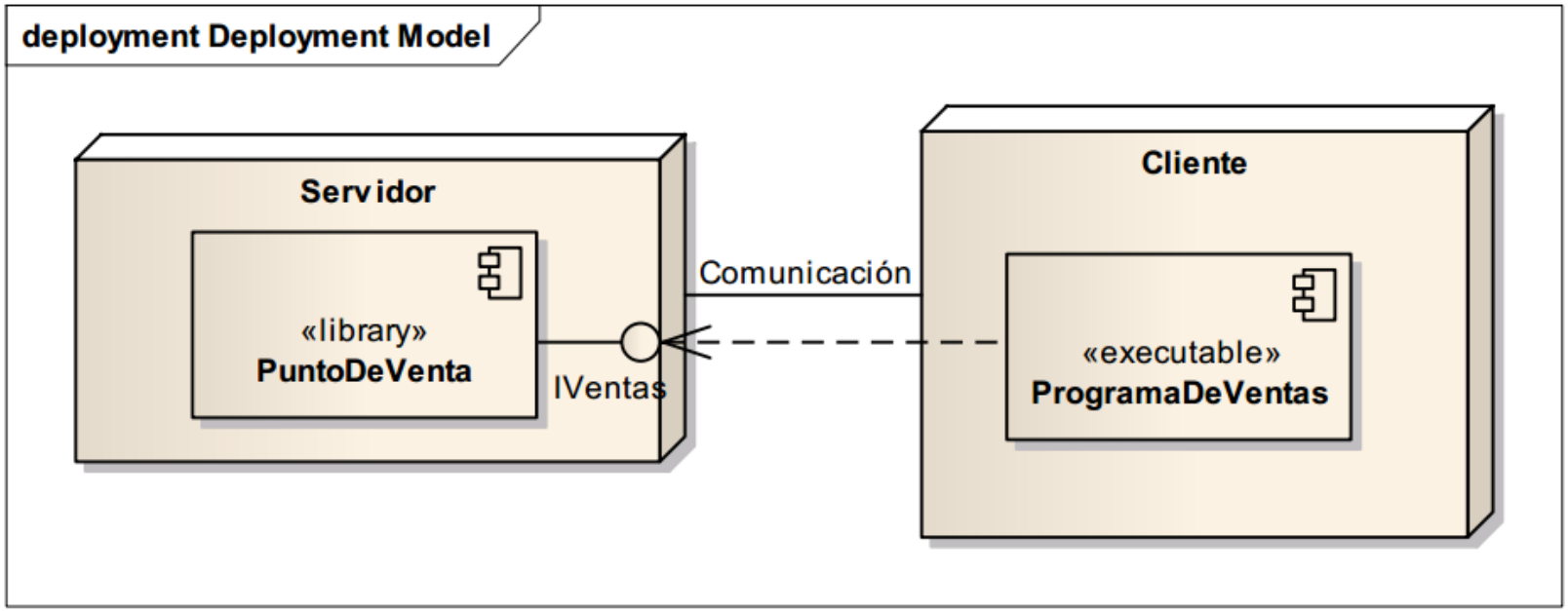

Firewall
AcceptRequest() : HTMLRequest
ForwardRequest() : HTMLRequest
ReturnResponse() : HTMLResponse
XSL Stylesheets
TransformXML()
Web Server
DoRequest() : HTMLResponse
ASP Pages
COM+ Component Environment
BusinessLogic
Oracle Database
ProcessSQLRequest() : Recordset

Diagrama de despliegue

- Muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos



Nodos

- Los nodos son los elementos donde se ejecutan los componentes.