

Diagramas de implementación

Maximiliano garrett

del castillo manuel

maximiliano diaz

nicolas diaz

roman basilico

cristina zegarra



**Diagramas de implementación**

**Contenido**

[**Introducción** 2](#_Toc528089828)

[**Diagramas de componentes** 2](#_Toc528089829)

[**Elementos de los diagramas de componentes** 3](#_Toc528089830)

[**Nodos** 9](#_Toc528089831)

[**Despliegue** 9](#_Toc528089832)

[**Diagramas de despliegue** 10](#_Toc528089833)

# **Introducción**

En el contexto de Lenguaje Unificado de Modelado (UML), un diagrama de implementación se encuentra dentro de la familia de diagramas estructurales porque describe un aspecto del sistema en sí. En este caso, el diagrama de implementación describe la implementación física de la información generada por el programa de software en los componentes de hardware. A la información que el software genera se la conoce como artefacto.

Los diagramas de implementación muestran la estructura del sistema en tiempo de ejecución. En un diagrama de este tipo se puede conocer cómo se configurarán e implementarán los elementos de hardware y software que forman una aplicación. Permiten una visión general del sistema sin entrar en detalles de implementación y comportamiento.

A continuación, veremos dos tipos de diagramas de implementación:

* Diagramas de componentes
* Diagramas de despliegue

# **Diagramas de componentes**

Los diagramas de componentes se utilizan para mostrar los diferentes componentes software existentes, así como su organización, sus interfaces y sus relaciones. Un diagrama de componentes representa las dependencias entre componentes software.

Algunos componentes existen en tiempo de compilación, otros en tiempo de enlace, otros en tiempo de ejecución y algunos en varias de estas instancias.

No es necesario que un diagrama de componentes represente a todo el sistema, suelen hacerse por partes y cada diagrama describe un apartado del sistema.

Si bien los diagramas de componentes prevalecen en la arquitectura de software, pueden ser utilizados para modelar y documentar cualquier arquitectura de sistema.

Utilidad:

Se utilizan para modelar la vista estática de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. Puede servir para ver que componentes pueden compartirse entre sistemas o entre diferentes partes del sistema.

## **Elementos de los diagramas de componentes**

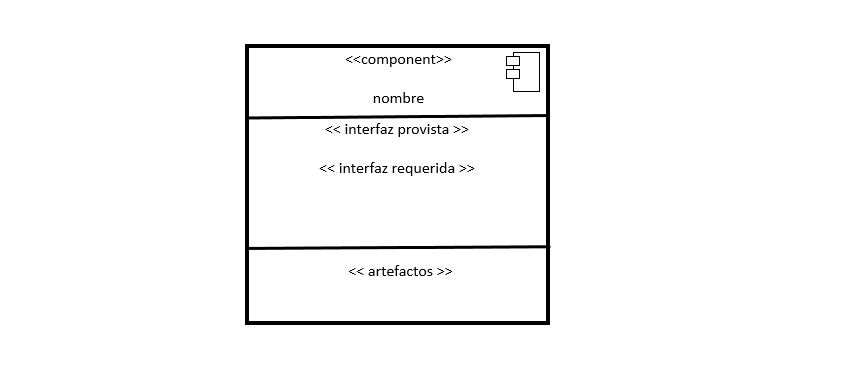
**Componente:**

parte física de un sistema (archivos, cabeceras, bibliotecas, módulos, ejecutables o paquetes). Cada componente debe tener un nombre que lo distinga de los demás. El componente es autónomo y encapsulado, esto permite su reutilización y reemplazo. Provee una o varias interfaces que le permiten interactuar o proveer servicios a otros componentes.

Notación:



Al igual que las clases, un componente puede enriquecerse con compartimentos adicionales que muestren sus detalles.



Propiedades de los componentes:

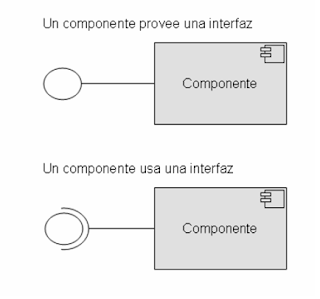
* Es una parte de un sistema
* Es reemplazable
* Conforma proporciona la realización de un conjunto de interfaces
* Es una unidad de despliegue independiente
* Puede ser conectado con otros componentes
* Realiza una función bien definida
* Presupone una infraestructura tecnológica que se piensa utilizar

Los componentes pueden agruparse en paquetes y pueden organizarse mediante relaciones entre ellos como pueden ser: dependencia, generalización, asociación y realización.

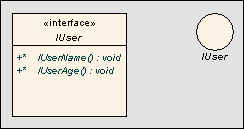
También pueden construirse a partir de otros componentes.

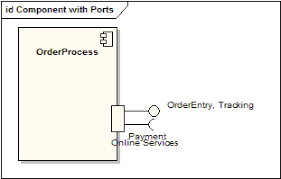
**Interfaz:**

Una interfaz es una colección de operaciones que especifican un servicio proporcionado o requerido por una clase o componente



Otra forma de representar la interfaz es por medio de una tarjeta similar a las clases, lo que permite dar mas detalles de la implementación:



**Puertos:**

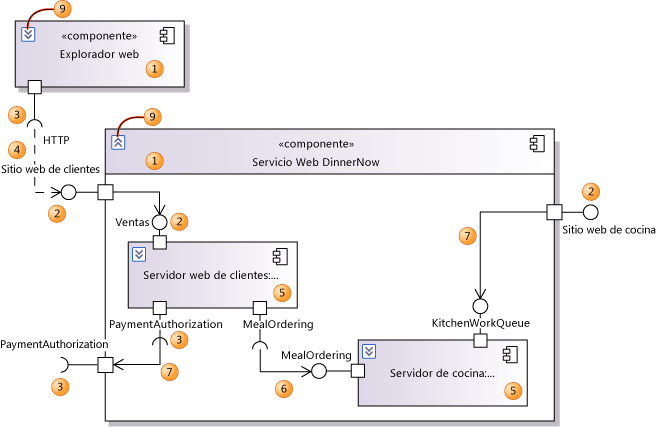
un puerto es una ventana explicita dentro de un componente encapsulado. Todas las interacciones dentro y fuera del componente encapsulado pasan a través de sus puertos. Representa un punto de interacción entre una instancia de clase o componente con su entorno o con las instancias que contiene (estructura interna). Todo puerto tiene un nombre.

Se representa con un cuadrado en el borde del componente desde donde se desprenden las interfaces.

**Estructura interna:**

la estructura interna de un componente esta formada por las partes que componen su implementación junto con las conexiones entre ellas. Las partes pueden ser conectadas a través de sus puertos

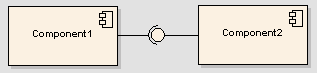
Cuando se muestra el funcionamiento interno del componente el puerto mapea las interfaces al interior del componente mostrando las relaciones de manera más clara.



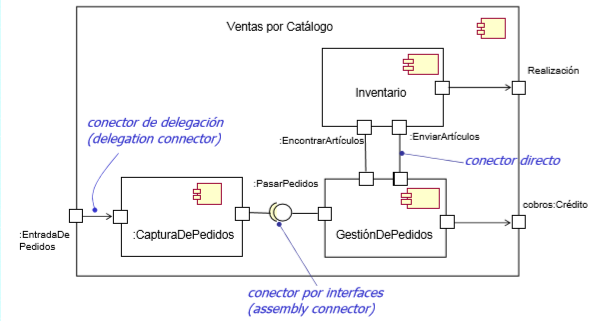
Una conexión entre dos puertos se denomina conector y denota un enlace en una instancia del componente. Los componentes pueden conectarse directamente o porque tienen interfaces compatibles.

**Conector de delegación:** conecta un puerto externo a uno interno.

**Ensamble:** es un conector que muestra justamente el ensamble que une una interfaz requerida de un componente con la interfaz proporcionada por otro componente.

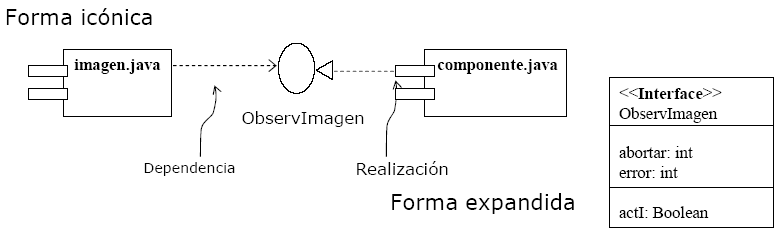


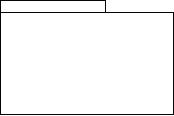
Ejemplo de conectores:



**Conectores de realización y dependencia**

**Realización:** se usa para mostrar que un componente realiza una interfaz

**Dependencia:** se utiliza para mostrar que un componente depende de otro componente o una interfaz

**Paquetes:**

la forma que tiene UML de agrupar elementos en subsistemas es a través de paquetes, pudiéndose anidar formando jerarquías de paquetes. Los paquetes agrupan piezas de software que se distribuye de manera conjunta.

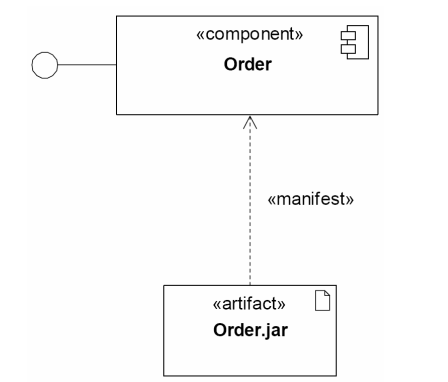
**Clase:**

es una representación de uno o más objetos que refleja su estructura y comportamiento en el sistema. Es la plantilla a partir de la cual se crean las instancias en ejecución.

**Artefacto:**

un artefacto es una implementación física de un conjunto de elementos lógicos como puede ser una clase. Representan el empaquetamiento físico de bits sobre la plataforma de implementación. Residen en nodos.

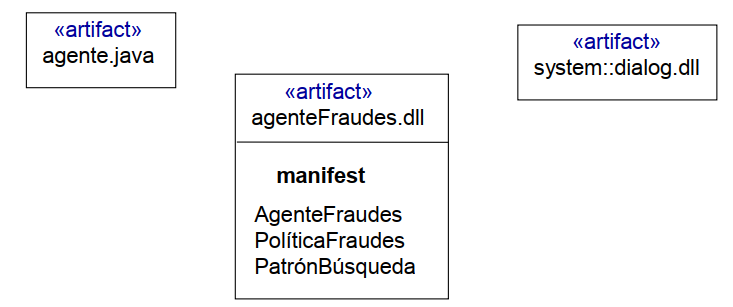
Un artefacto puede ser la implementación de una clase o un componente.



Existen básicamente 3 tipos de artefactos:

* Componentes de despliegue: componentes necesarios para formar un sistema ejecutable
* Componentes producto del trabajo: productos que quedan al final del proceso de desarrollo: código fuente, archivos de datos, etc.
* Componentes de ejecución: se crean durante la ejecución como un objeto .NET a partir de una librería DLL.

Notación de artefactos:



**Estereotipos para artefactos**

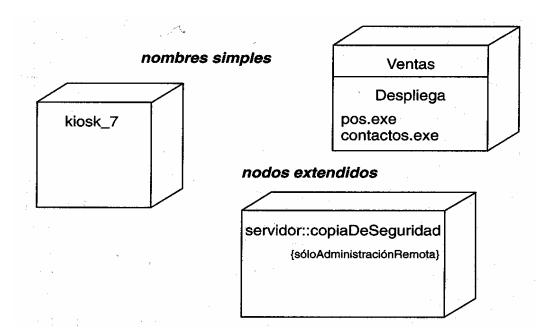
UML ofrece estereotipos predefinidos para los artefactos:

* **Document:** fichero genérico que no es código fuente o ejecutable. Subclase de file
* **Ejecutable:** artefacto que se puede ejecutar en un nodo. Subclase de file
* **File:** archivo físico del sistema desarrollado
* **Library:** fichero de una biblioteca de objetos estática o dinámica. Subclase de file
* **Source:** fichero de código fuente

# **Nodos**

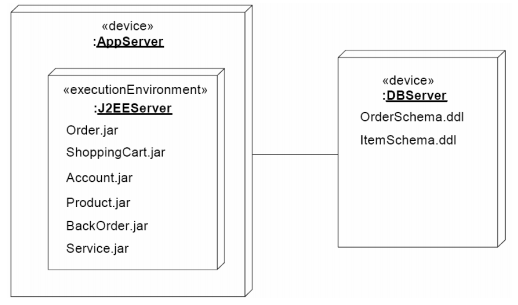
Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional con memoria y capacidad de procesamiento. Se utilizan para modelar la tipología del hardware sobre el que se ejecuta el sistema. Representa un dispositivo sobre el que se pueden desplegar componentes. Se pueden estereotipar y agrupar en paquetes. Pueden tener atributos y operaciones.

Notación: se representan con un prisma



Hay dos estereotipos predefinidos de nodo:

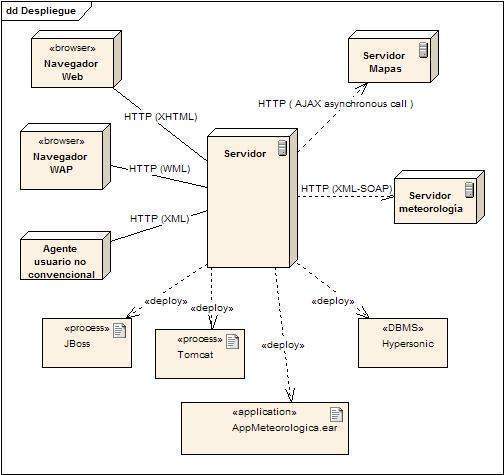
* **Unidad (<< device >>):** recurso computacional físico sobre el cual pueden ser desplegados artefactos para su ejecución
* **Entorno de ejecución (<< ExecutionEnviroment >>):** nodo que ofrece un entorno para ejecutar un tipo específico de artefactos ejecutables



# **Despliegue**

El despliegue es el proceso de asignar artefactos a nodos o instancias de artefactos a instancias de nodos. Para mostrar este despliegue se utilizan los diagramas de despliegue.

# **Diagramas de despliegue**



Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Estos diagramas son los complementos de diagrama de componentes que, unidos, proveen la vista de implementación del sistema. Describe la topología del sistema la estructura de los elementos de hardware y software que ejecuta cada uno de ellos.

Los nodos se interconectan mediante soportes bidireccionales. Que puede a su vez estereotiparse. Esta vista permite determinar las consecuencias de la distribución y la asignación de recursos. Las instancias de los nodos pueden contener instancias de ejecución, como instancias de componentes y objetos. El modelo puede mostrar dependencias entre las instancias y sus interfaces, y también modelar la migración de entidades entre nodos u otros contenedores. Esta vista tiene una forma de descriptor y otra de instancia. La forma de instancia muestra la localización de las instancias de los componentes específicos en instancias específicas del nodo como parte de una configuración del sistema. La forma de descriptor muestra que tipo de componentes pueden subsistir en que tipos de nodos y que tipos de nodos se pueden conectar, de forma similar a un diagrama de clases.

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos, procesos (caso particular de un objeto). En general un nodo será una unidad de computación de algún tipo, desde un sensor a un mainframe (computadora muy potente). Las instancias de componentes software pueden estar unidas por relaciones de dependencia, posiblemente a interfaces (ya que un componente puede tener más de una interfaz).

Cuando se utilizan los diagramas de despliegue

* Cuando se desarrolla un software que interactúa con dispositivos que normalmente no gestiona el sistema operativo
* El sistema está distribuido físicamente sobre varios procesadores
* Es necesario razonar sobre la topología de dispositivos sobre los que se ejecuta el software

Dentro de los diagramas de despliegue nos vamos a encontrar con los siguientes elementos:

* Nodos
* Relaciones de dependencia y asociación entre nodos

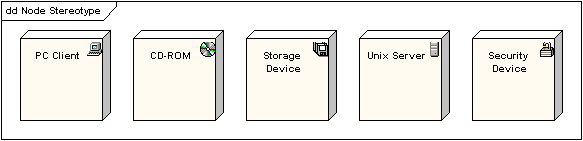
Opcionalmente puede contener:

* Notas y restricciones
* Artefactos, cada uno dentro de un nodo
* Paquetes y subsistemas
* Instancias (para visualizar un caso concreto de una familia de topologías de hardware)

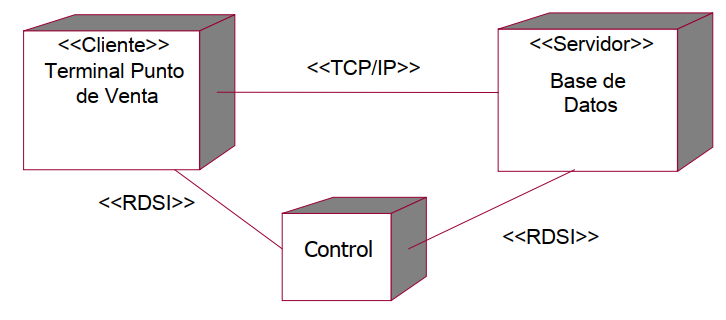
**Estereotipo de nodo**

Un numero de estereotipos estandar se provee para los nodos nombrados

<<cdrom>> <<cdrom>> <<computer>> <<diskarray>> <<pc>> <<pccomputer>> <<pcserver>> <<secure>> <<server>> <<Storage>> <<unix server>> <<user pc>>.

Estos mostraran un icono apropiado en la esquina derecha aariba del simbolo del nodo

Notación con uso de estereotipos para distinguir nodos



**Asociación:** en el contexto del diagrama de despliegue una asociación representa una ruta de comunicación entre nodos.

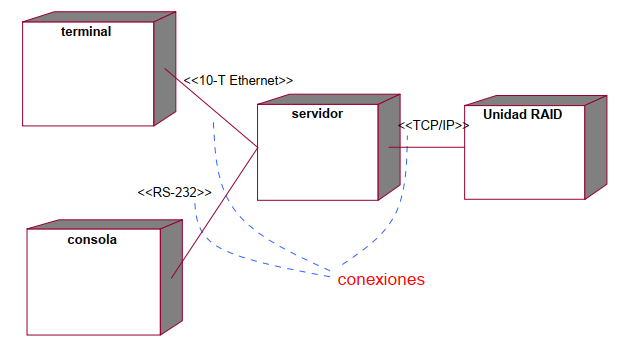


Diagrama donde se muestra como se despliegan los componentes dentro de los nodos

