

Fundamentos de Ingeniería de Software

Marcello Visconti y Hernán Astudillo
Departamento de Informática
Universidad Técnica Federico Santa María
{visconti,hernan} en inf.utfsm.cl

Aspectos Avanzados de Diseño

Contenido



- ✍ Arquitectura multi-capas
- ✍ Paquetes de software
- ✍ Patrones avanzados

Aspectos Avanzados de Diseño

Introducción



- ✍ En el estudio anterior de casos se enfocó en los objetos del dominio del problema (entre ellos, "Venta"), porque así se definen los conceptos y el comportamiento básicos de un sistema.
- ✍ Pero un sistema se compone de muchos subsistemas, uno de los cuales son los objetos del dominio.
- ✍ Un sistema ordinario de información ha de conectarse a la interfaz del usuario y a un mecanismo de almacenamiento persistente.

Aspectos Avanzados de Diseño

Arquitectura clásica de 3 capas [1]

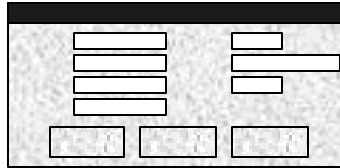


- ✍ Una arquitectura común de los sistemas de información que abarca una interfaz para el usuario y el almacenamiento persistente de datos se conoce con el nombre de arquitectura de tres capas:
 - ✍ *Presentación*: ventanas, reportes, etétera.
 - ✍ *Lógica de aplicaciones*: tareas y reglas que rigen el proceso.
 - ✍ *Almacenamiento*: mecanismo de almacenamiento persistente.

Aspectos Avanzados de Diseño

Arquitectura clásica de 3 capas [2]

Presentación

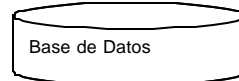


Lógica de aplicaciones

Registrar
Ventas

Autorizar
pagos

Almacenamiento



Fundamentos de Ingeniería de SW

5

Aspectos Avanzados de Diseño

Arquitectura clásica de 3 capas [3]

- ✍ La calidad tan especial de la arquitectura de tres capas consiste en aislar la lógica de la aplicación y en convertirla en una capa intermedia bien definida y lógica del software
- ✍ En la capa de presentación se realiza relativamente poco procesamiento de la aplicación
 - ✍ Las ventanas envían a la capa intermedia peticiones de trabajo y éste se comunica con la capa de almacenamiento del extremo posterior
 - ✍ Esta arquitectura contrasta con el diseño de dos capas, donde - por ejemplo - colocamos la lógica de aplicaciones dentro de las definiciones de ventana, que leen y escriben directamente en una base de datos; no hay una capa intermedia que separe la lógica

Fundamentos de Ingeniería de SW

6

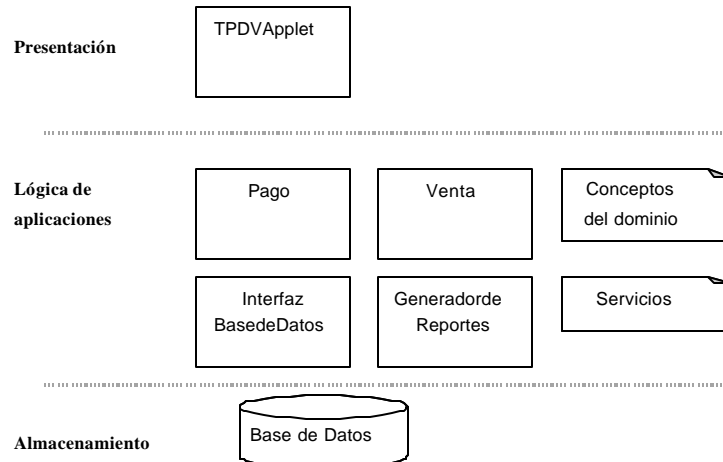
Aspectos Avanzados de Diseño

Descomposición de la capa de lógica [1]

- ✎ En un diseño orientado a objetos, la capa de la lógica de aplicaciones se divide en otras menos densas.
- ✎ La capa de la lógica de aplicaciones está constituida por las siguientes capas:
 - ✎ *Objetos del dominio*: clases que representan los conceptos del dominio; por ejemplo, una venta.
 - ✎ *Servicios*: los objetos servicio de las funciones como la interacción de bases de datos, los reportes, las comunicaciones y la seguridad.

Aspectos Avanzados de Diseño

Descomposición de la capa de lógica [2]



Aspectos Avanzados de Diseño

Descomposición de la capa de lógica [3]

- ✎ Entre los motivos por los cuales se recurre a la arquitectura multicapas se cuentan los siguientes:
 - ✎ Aislamiento de la lógica de aplicaciones en componentes independientes susceptibles de reutilizarse después en otros sistemas.
 - ✎ Distribución de las capas en varios nodos físicos de cómputo y en varios procesos.
 - ✎ Esto puede mejorar el desempeño, la coordinación y el compartir la información en un sistema de cliente-servidor.
 - ✎ Asignación de los diseñadores para que construyan determinadas capas; por ejemplo, un equipo que trabaje exclusivamente en la capa de presentación.
 - ✎ Y así se brinda soporte a los conocimientos especializados en las habilidades de desarrollo y también a la capacidad de realizar actividades simultáneas en equipo.

Aspectos Avanzados de Diseño

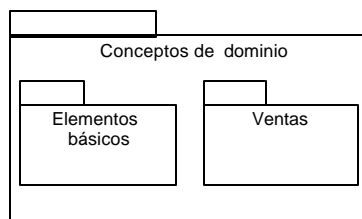
Paquetes de UML [1]

- ✎ UML ofrece el mecanismo *paquete* que permite describir los grupos de elementos o subsistemas.
 - ✎ Un paquete es un conjunto de cualquier tipo de elementos de un modelo: clases, casos de uso, diagramas de colaboración u otros paquetes (los anidados).
- ✎ Un sistema puede examinarse íntegramente dentro del ámbito de un sólo paquete de alto nivel: *el paquete Sistema*.
- ✎ El paquete define un espacio de un nombre anidado, de modo que los elementos del mismo nombre pueden duplicarse dentro de varios paquetes.

Aspectos Avanzados de Diseño

Paquetes de UML [2]

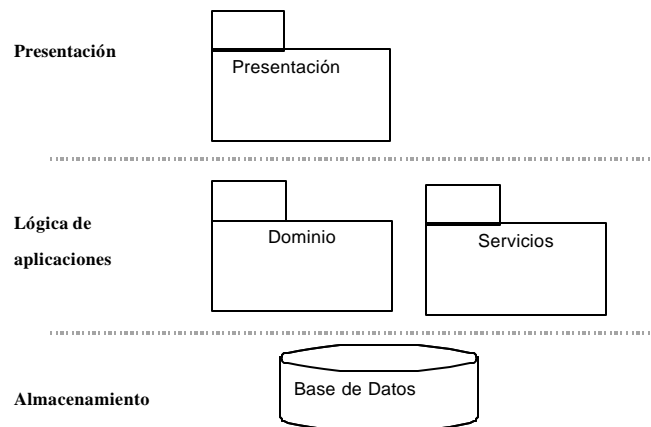
- ✍ Un paquete se muestra gráficamente como una carpeta con etiquetas. Los paquetes subordinados se incluyen en su interior.
- ✍ El nombre del paquete se encuentra dentro de la etiqueta si el paquete describe sus elementos; en caso contrario estará en el centro de la misma carpeta.



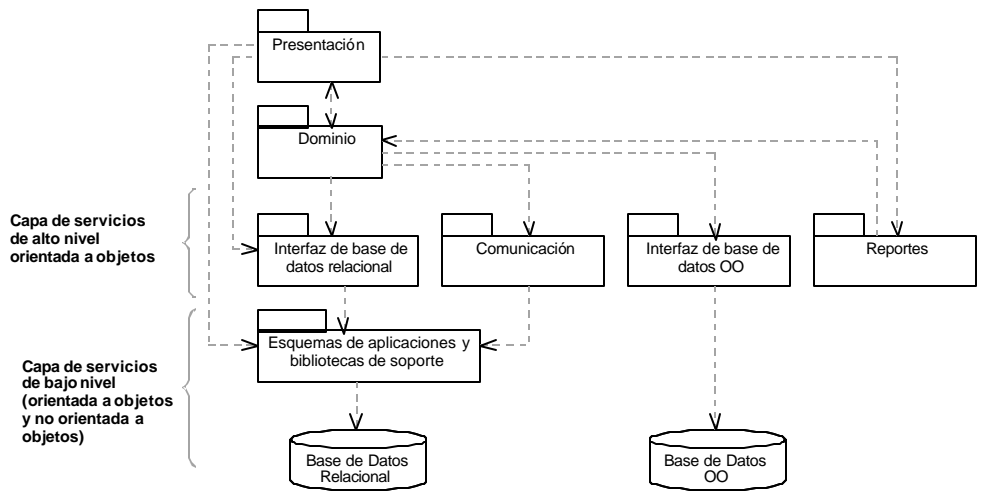
Aspectos Avanzados de Diseño

Paquetes de UML [3]

- ✍ Diagramas de Arquitectura



Aspectos Avanzados de Diseño Arquitectura detallada [1]



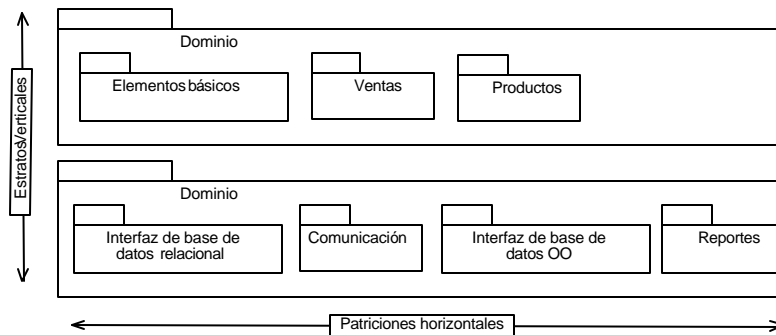
Aspectos Avanzados de Diseño Arquitectura detallada [2]

- ⌘ Las relaciones de dependencia (línea punteada con flecha) indican si un paquete tiene acoplamiento con otro.
 - ⌘ La ausencia de dependencia del paquete *A* al *B* significa que los componentes de *A* no tienen referencias a ninguna clase, componente, interfaz, método o servicio de *B*.
- ⌘ Agrupe los elementos en un paquete aplicando la siguiente directriz:
 - ⌘ Agrupe los elementos para ofrecer en un paquete un servicio común (o una familia de servicios relacionados), con un nivel relativamente alto de acoplamiento y colaboración.
 - ⌘ En cierto nivel de abstracción, se verá el paquete como muy cohesivo (tiene responsabilidades estrechamente relacionadas).
 - ⌘ En cambio, habrá relativamente bajo acoplamiento y colaboración entre los elementos de los paquetes.

Aspectos Avanzados de Diseño

Estratos y particiones

- Podemos caracterizar una arquitectura multicapas como compuesta de estratos (subcapas) y particiones.
- Los estratos de una arquitectura representan las capas verticales, mientras que las particiones representan la división horizontal de subsistemas relativamente paralelos de un estrato.



Fundamentos de Ingeniería de SW

15

Aspectos Avanzados de Diseño

Visibilidad entre las clases de paquetes [1]

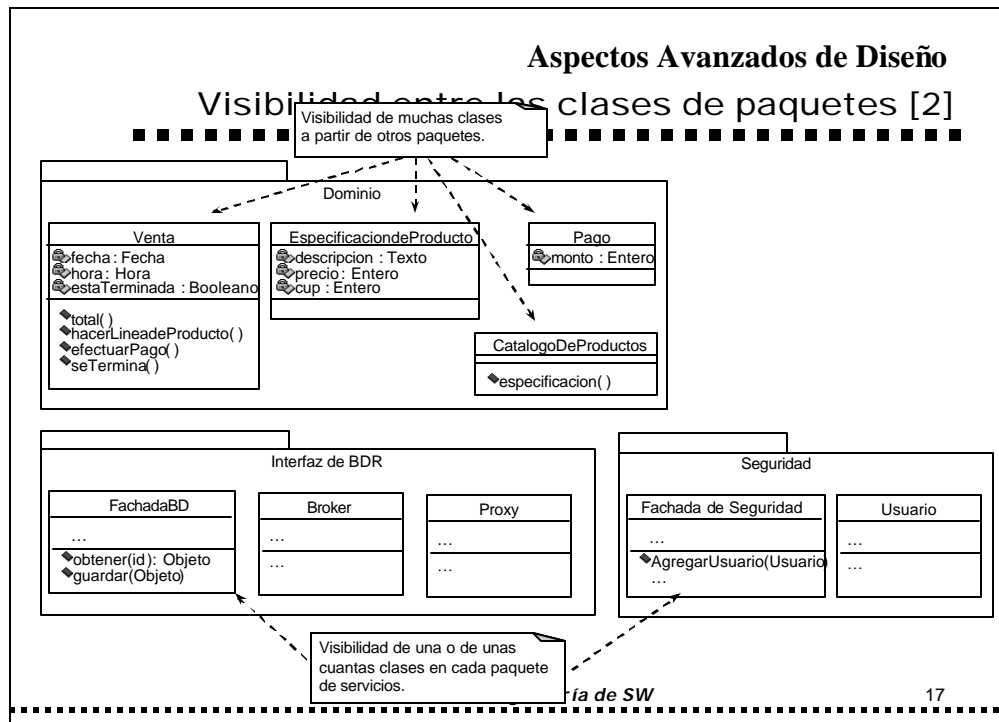
- La visibilidad respecto a las clases en varios paquetes suele conformar el siguiente patrón:
 - Acceso a los paquetes del dominio*: otros paquetes, generalmente el de presentación, tienen acceso a la visibilidad de muchas clases que representan los conceptos del dominio.
 - Acceso a los paquetes de servicios*: otros paquetes, generalmente los de dominio y de presentación, tienen acceso a la visibilidad sólo respecto a una clase o unas cuantas en determinado paquete de servicios (casi siempre una clase Fachada).
 - Acceso a los paquetes de presentación*: ningún otro paquete tendrá acceso directo a la visibilidad de la capa de presentación. La comunicación es indirecta (mediante el patrón "Observador"), si es que existe.

Fundamentos de Ingeniería de SW

16

Aspectos Avanzados de Diseño

Visibilidad entre las clases de paquetes [2]



Aspectos Avanzados de Diseño

Patrones de Diseño: Fachada

- ¿Cómo deberían los componentes coordinarse con las clases de un paquete de servicios - por ejemplo, el paquete *InterfazdeBasedeDatosRelacional*, que ofrece servicios para realizar transformaciones entre los objetos y los renglones de las tablas?
- Cuando se define una clase con una interfaz común a un grupo de componentes o a un conjunto heterogéneo de interfaces, a esa clase le damos el nombre de Fachada.
- Los elementos heterogéneos pueden ser las clases de un paquete, un conjunto de funciones, un esquema o un subsistema local o remoto.

Aspectos Avanzados de Diseño

Patrones de Diseño: Fachada

- ✎ **Problema:** Se requiere una interfaz común y unificada para un conjunto heterogéneo de interfaces, un subsistema por ejemplo. ¿Qué hacer?
- ✎ **Solución:** Definir una clase individual que unifique la interfaz y le asigne la responsabilidad de colaborar con el subsistema. El patrón Fachada a menudo sirve para suministrar una interfaz pública a un paquete de servicios.
 - ✎ Por ejemplo, si se dispone de un paquete de *InterfazBasededatosRelacional* con muchas clases internas, podemos definir una clase como *FachadaBD* que ofrezca la interfaz pública común con los servicios del paquete. Las clases de otros paquetes envían mensajes únicamente a una instancia de ella y no se acoplan a otras clases del paquete.

Aspectos Avanzados de Diseño

Patrones de Diseño: Separación Modelo-Vista

- ✎ El patrón de "Separación Modelo-Vista" establece que los objetos modelo (dominio) no deberían conocer directamente los objetos vista (presentación) ni estar directamente acoplados a ellos.
 - ✎ Los métodos de una clase de modelo no deberían contener instrucciones que envíen mensajes directos a un objeto vista; este tipo de clase no debería conocer las interfaces del usuario ni relacionarse con ellas mediante códigos.
 - ✎ Una clase de modelos no tendrá visibilidad directa respecto a una clase vista. Pero se admite la visibilidad indirecta.

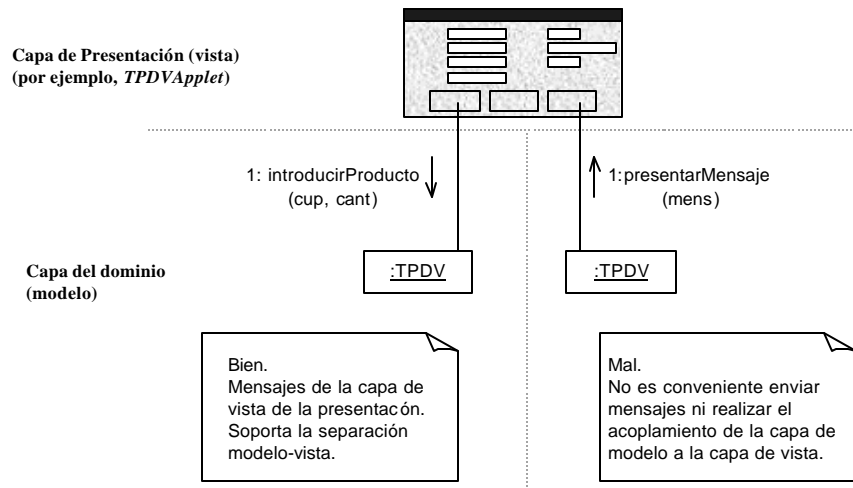
Aspectos Avanzados de Diseño

Patrones de Diseño: Separación Modelo-Vista

- ✍ **Contexto/problema:** Conviene desacoplar los objetos dominio (modelo) y las ventanas (vistas), a fin de brindar soporte a un mayor reuso de los objetos dominio y reducir al mínimo el impacto que los cambios de la interfaz tienen en ellos. ¿Qué hacer?
- ✍ **Solución:** Definir las clases de dominio (modelo) para que no tengan acoplamiento ni visibilidad directa respecto a las clases ventana (vista) y para que los datos de la aplicación y de la funcionalidad se conserven en las clases de dominio, no en las de ventana.

Aspectos Avanzados de Diseño

Patrones de Diseño: Separación Modelo-Vista



Aspectos Avanzados de Diseño

Patrones de Diseño: Separación Modelo-Vista

- ✍ Entre las razones por las cuales se emplea el patrón de separación Modelo-Vista, podemos citar las siguientes:
 - ✍ Dar soporte a las definiciones de modelos cohesivos que se centran en los procesos de dominio más que en las interfaces de computadora.
 - ✍ Permitir desarrollar independientemente el modelo y las capas de interfaz para el usuario.
 - ✍ Reducir al mínimo el impacto que los cambios de requerimientos de la interfaz tienen en la capa de dominio.
 - ✍ Permitir conectar fácilmente otras vistas a la capa actual de dominio, sin que esto la afecte.

Aspectos Avanzados de Diseño

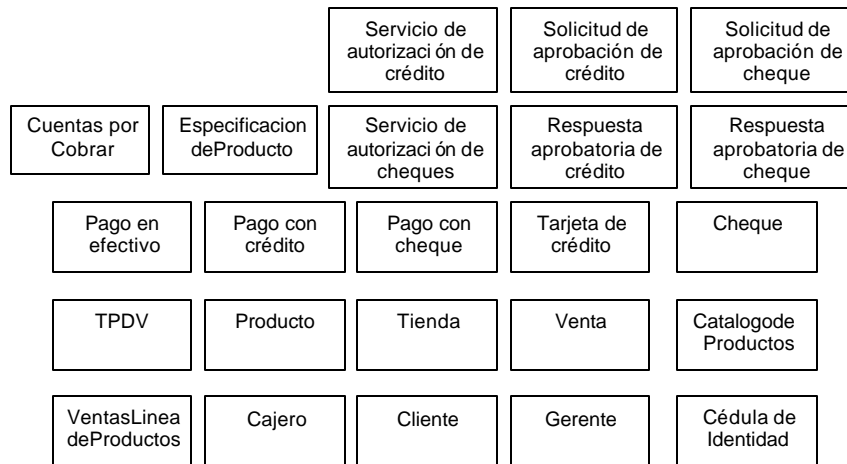
Patrones de Diseño: Separación Modelo-Vista

- ✍ Entre las razones por las cuales se emplea el patrón de separación Modelo-Vista, podemos citar las siguientes:
 - ✍ Permitir vistas simultáneas y múltiples del mismo objeto modelo; por ejemplo, la vista de la estadística de ventas en un diagrama tabular.
 - ✍ Permitir ejecutar la capa del modelo independiente de la capa de interfaz para el usuario; por ejemplo, en un sistema de procesamiento de mensajes o de modo lotes.
 - ✍ Permitir transportar fácilmente la capa de modelo a otro esquema de interfaz para el usuario.

Aspectos Avanzados de Diseño

Paquetes

Punto de Venta



Aspectos Avanzados de Diseño

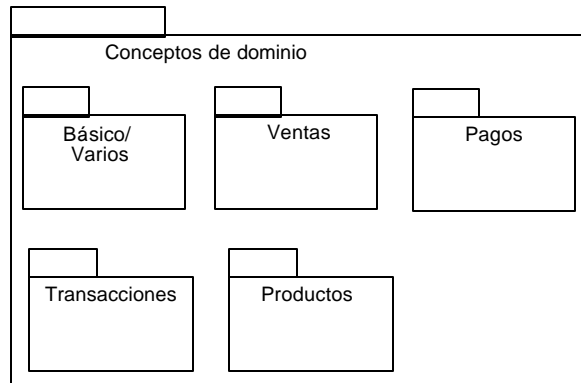
Paquetes

Punto de Venta

- ⌘ Agrupe los elementos en un paquete aplicando la siguiente directriz:
 - ⌘ Agrupe los elementos para ofrecer en un paquete un servicio común (o una familia de servicios relacionados), con un nivel relativamente alto de acoplamiento y colaboración.
 - ⌘ En cierto nivel de abstracción, se verá el paquete como muy cohesivo (tiene responsabilidades estrechamente relacionadas).
 - ⌘ En cambio, habrá relativamente bajo acoplamiento y colaboración entre los elementos de los paquetes.

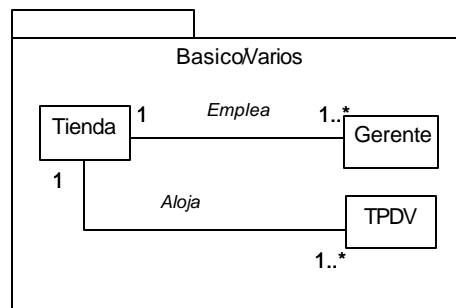
Aspectos Avanzados de Diseño

Paquetes Punto de Venta



Aspectos Avanzados de Diseño

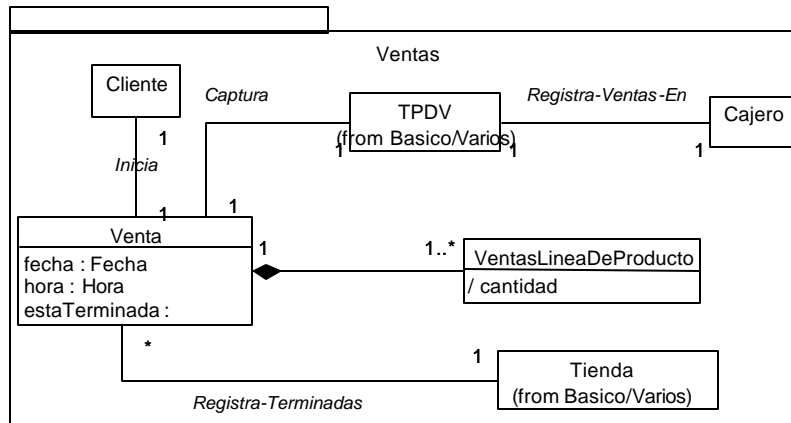
Paquetes Punto de Venta



Aspectos Avanzados de Diseño

Paquetes

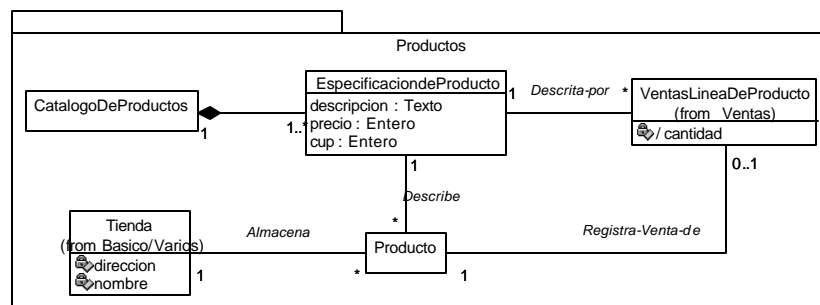
Punto de Venta



Aspectos Avanzados de Diseño

Paquetes

Punto de Venta



Aspectos Avanzados de Diseño

Resumen



- ✍ Arquitectura multi-capas
- ✍ Paquetes de software

Aspectos Avanzados de Diseño

Quiz



- ✍ ¿Qué es una arquitectura de tres o más capas?
- ✍ ¿Qué beneficios ofrece?
- ✍ ¿Qué es un paquete de software?
- ✍ Describa la relación entre paquetes y arquitectura multicapa.
- ✍ ¿Cuándo se dan dependencias entre paquetes?
- ✍ ¿Cómo se generan los paquetes?