



## LOGRO DE LA SESIÓN



*Al término de la sesión de aprendizaje el estudiante elabora el documento de arquitectura de Software plasmando en una especificación formal el diseño y arquitectura fundamental del producto.*



- ❖ Modelo de Diseño
- ❖ Extensiones UML para aplicaciones WEB (WAE).
- ❖ Elaboración del documento de Arquitectura de Software.
- ❖ Actividad de aprendizaje.
- ❖ Evaluación T3



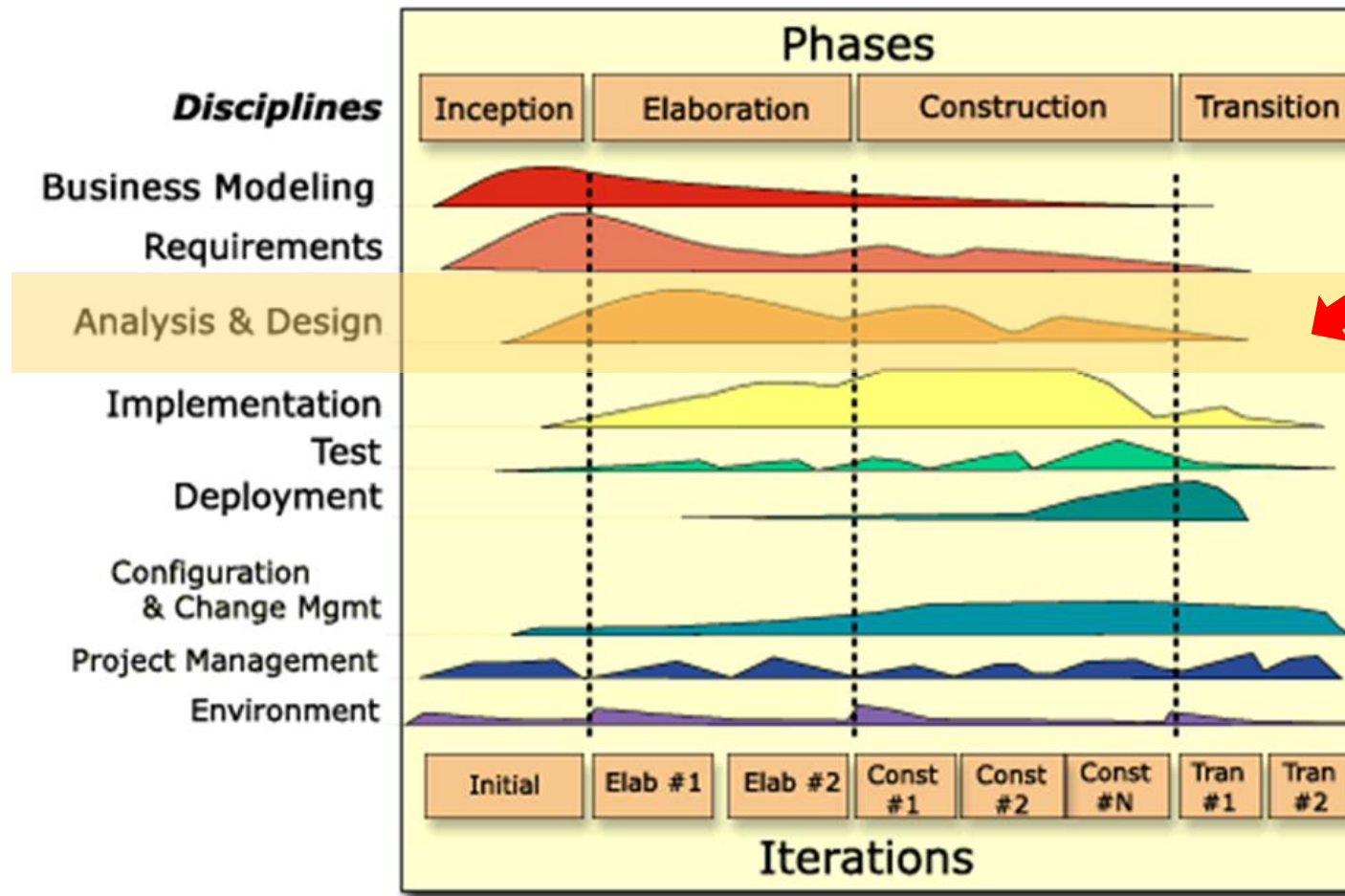


# Modelo de Diseño

**Implementación del Modelo de  
Diseño**

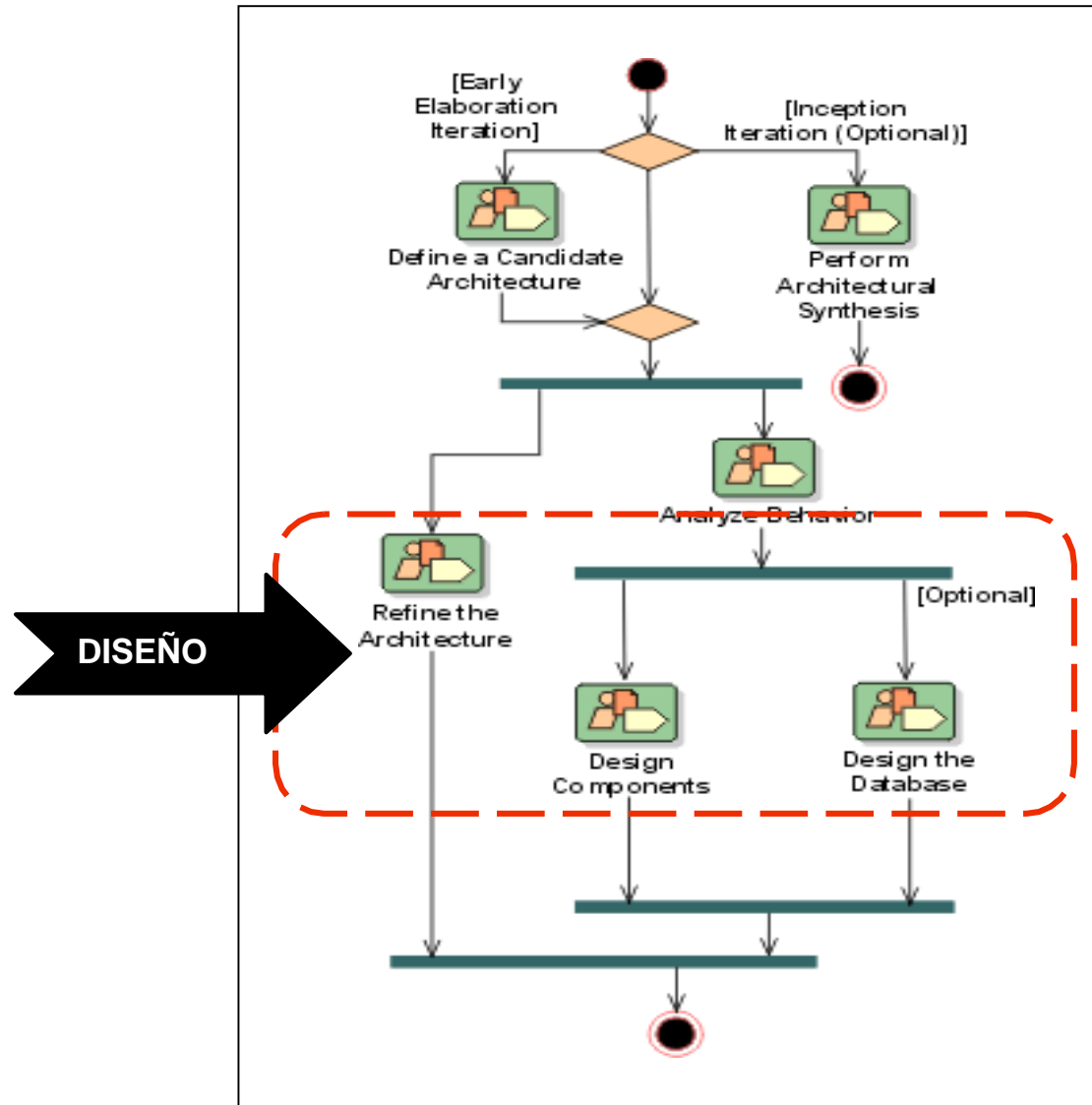
# DISCIPLINA ANÁLISIS Y DISEÑO (RUP)

## Tercera Disciplina



# DISCIPLINA ANÁLISIS Y DISEÑO (RUP)

## Flujo de Trabajo del Análisis y Diseño

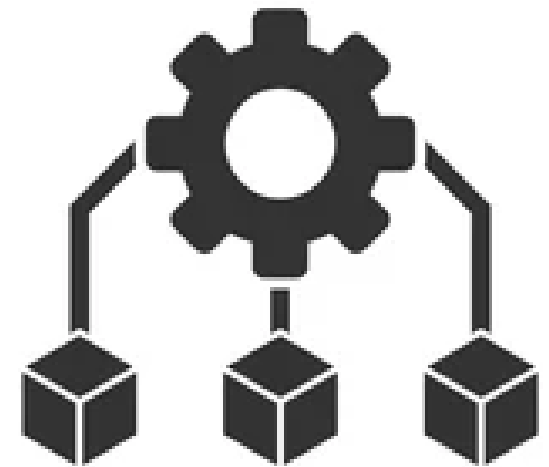


# Modelo de Diseño

## Flujo de Trabajo del diseño Orientado a Objetos



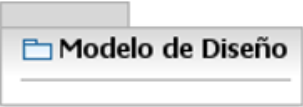
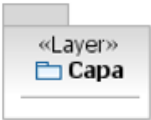
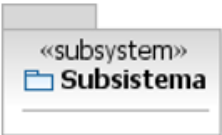

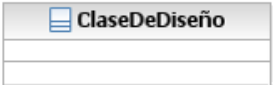
El objetivo del **diseño** es **entender la solución** refinando el modelo de análisis con la intención de desarrollar un modelo de diseño que permita una **transición** sin problemas a la **fase de construcción**. En el diseño, nos adaptamos al entorno de implementación y despliegue.



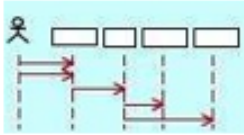


# Artefactos del Diseño

## Artefactos



Artefacto	Descripción
 Modelo de Diseño	Representa la vista interna del sistema. Conjunto formado por las clases de diseño y las realizaciones de casos de uso. El modelo de diseño se convertirá en la materia prima que nuestra disciplina de implementación transformará en código ejecutable.
 Capa	Representan un medio para organizar los artefactos del modelo de diseño. Dependiendo del estilo arquitectónico, una capa agrupa un conjunto de subsistemas junto con sus clases de diseño.
 Subsistema	Un subsistema contiene las clases de diseño de un grupo de casos de uso. Tiene correspondencia directa con los paquetes de análisis.
 Librería	Una librería contiene clases utilitarias, adicionales a las del API del lenguaje de programación, implementadas por el equipo de desarrollo.
 Clases de diseño	Son abstracciones de clase directamente utilizadas en la implementación, es decir, estas clases junto con sus atributos y operaciones se mapean directamente en el lenguaje de programación.

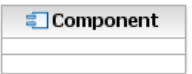
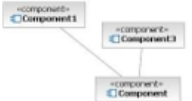
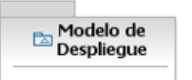


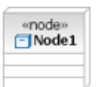
 Realización de Diseño de Caso de Uso	Describe cómo un caso de uso se lleva a cabo en términos de clases de diseño y sus objetos. Hay una correspondencia directa entre Realización de Diseño de Casos de Uso y Realización de Análisis de Casos de Uso.
 Diagrama de Clases	El diagrama de clases describe la estructura de un caso de uso. Contiene las clases que participan en el caso de uso, aunque algunas de ellas puedan participar en varios.
 Diagramas de Secuencia	Muestra una secuencia detallada de interacción entre los objetos de diseño. Visualizan el intercambio de mensajes entre objetos. Se crea un diagrama de secuencia por cada flujo de trabajo del caso de uso: flujo básico, subflujos y flujos alternativos.



# Artefactos del Diseño

## Artefactos (Continuación)



Artefacto	Descripción
 Componente	Un componente representa una pieza del software reutilizable. Por ejemplo, si se está desarrollando una aplicación web estas piezas pueden ser recursos estáticos (páginas HTML) y recursos dinámicos (JSP y <del>servlets</del> <b>Servlets</b> ) representados como componentes.
 Diagrama de componentes	Un diagrama de componentes muestra la estructura de un sistema <i>software</i> , el cual describe los componentes <i>software</i> , sus interfaces y sus dependencias.
 Modelo de Despliegue	Describe la distribución física del sistema en términos de cómo las funcionalidades se distribuyen entre los nodos de computación sobre los que se va a instalar el sistema.
 Diagrama de Despliegue	Un diagrama de despliegue se puede utilizar para visualizar la topología actual del sistema y la distribución de componentes.
 Artefacto	Los artefactos son elementos que representan las entidades físicas de un sistema <i>software</i> . Los artefactos representan unidades de implementación física como archivos ejecutables, librerías, componentes de <i>software</i> , documentos, y bases de datos.
 Nodo	Representa un recurso de computación. Los nodos tienen relaciones entre ellos que representan los medios de comunicación que hay entre éstos como una Intranet o Internet. La funcionalidad de un nodo viene representada por los componentes que se ejecutan en él.

# Modelo de Diseño

## Modelo de Análisis Vs Modelo de diseño




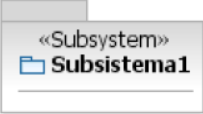

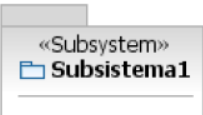

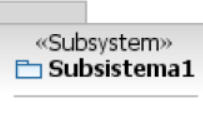

<i><b>Modelo de Análisis</b></i>	<i><b>Modelo de Diseño</b></i>
Es un modelo conceptual y genérico, es una abstracción del sistema.	Es un modelo físico y concreto, es un plano de la implementación.
Es menos formal.	Es más formal.
Es un bosquejo del diseño del sistema.	Es una realización del diseño del sistema.
Puede no mantenerse durante todo el ciclo de vida del <i>software</i> .	Debe ser mantenido durante todo el ciclo de vida del <i>software</i> .
Define una estructura para modelar el sistema.	Da forma al sistema.

# Modelo de Diseño

## Capas Lógicas de la Arquitectura



### Capas, subsistemas, librerías y elementos de diseño según patrón arquitectónico MVC

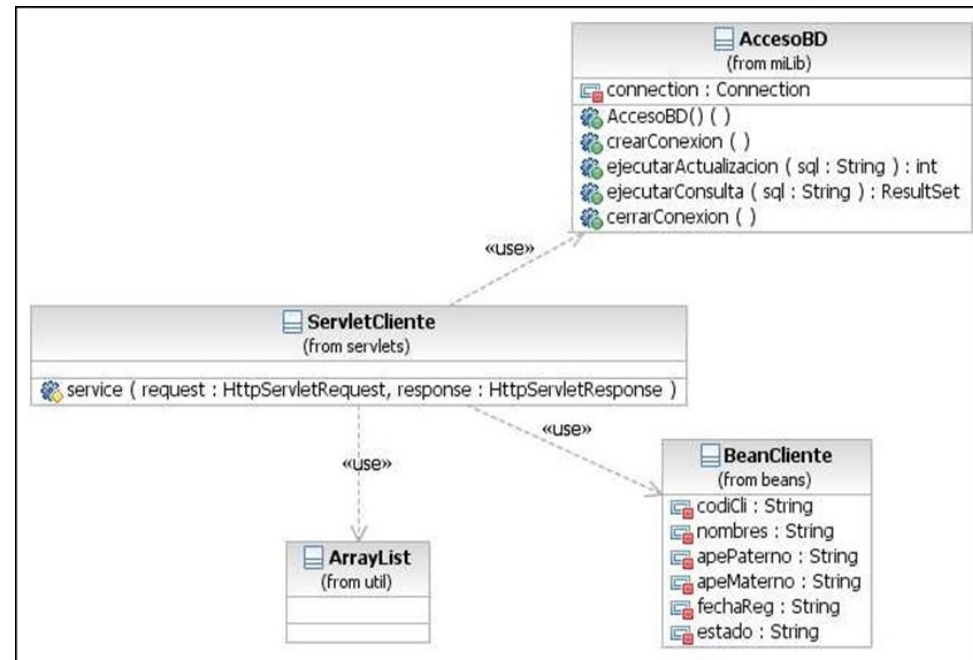
Capa	Subsistema/Librerías	Componentes
	 	Clases estereotipadas: <ul style="list-style-type: none"><li>• Páginas HTML: &lt;&lt;Client Page&gt;&gt; y &lt;&lt;HTML Form&gt;&gt;</li><li>• Páginas JSP: &lt;&lt;Server Page&gt;&gt;, &lt;&lt;Client Page&gt;&gt; y &lt;&lt;HTML Form&gt;&gt;</li></ul>
		Clase estereotipada para servlets: <<Http Servlet>>
		Clases de diseño: <i>beans</i> .
		Clases de diseño: clases utilitarias.

# Realizaciones de Diseño

## Diagrama de Clases de Diseño



Es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.





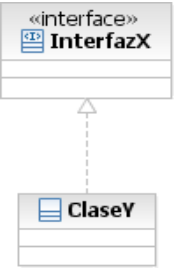
# Realizaciones de Diseño

## Diagrama de Clases de Diseño: Relacione



Las relaciones que pueden existir entre clases de diseño son:

Herencia e implementación

Tipo de relación	UML	Java
Herencia		<pre>public class ClaseA {     //Más código }  public class ClaseB extends ClaseA {     //Más código }</pre>
Implementación		<pre>public interface InterfazX {     //Más código }</pre>
		<pre>public class ClaseY implements InterfazX {     //Más código }</pre>




# Realizaciones de Diseño

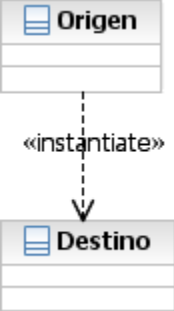
## Diagrama de Clases de Diseño: Relacione



Las relaciones que pueden existir entre clases de diseño son:

Relaciones de dependencia

Tipo de dependencia	UML	Descripción
<b>&lt;&lt;use&gt;&gt;</b> (De uso)	 <pre>graph TD; Origen[Origen] -.-&gt; «use»  Destino[Destino];</pre>	El funcionamiento del origen depende del funcionamiento del destino.




# Realizaciones de Diseño

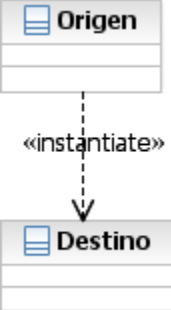
## Diagrama de Clases de Diseño: Relacione



Las relaciones que pueden existir entre clases de diseño son:

Relaciones de  
dependencia

Tipo de dependencia	UML	Descripción
<<instantiate>> (De instancia)		El origen solo crea instancias del destino.

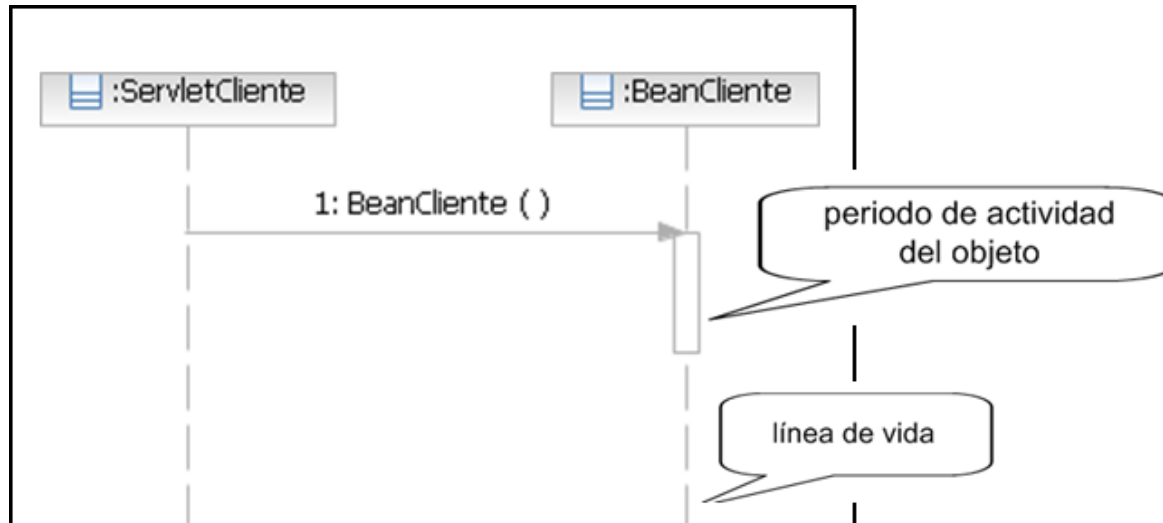


# Realizaciones de Diseño

## Diagramas de Secuencia



El diagrama de secuencia describe la dinámica del sistema, describiendo las interacciones entre un grupo de objetos mostrando de forma secuencial los envíos de mensajes entre objetos. El diagrama puede asimismo mostrar los flujos de datos intercambiados durante el envío de mensajes.





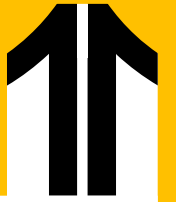


# Modelo de Diseño

**Extensiones UML para Aplicaciones  
WEB**

# Extensiones de UML para aplicaciones Web.

## ¿Qué es un aplicativo web?



Una aplicación Web es un sitio Web donde la navegación a través del sitio, y la entrada de datos por parte de un usuario, afectan el estado de la lógica del negocio. En esencia, una aplicación Web usa un sitio Web como entrada (front-end) a una aplicación típica. ...Si no existe lógica del negocio en el servidor, el sistema no puede ser llamado aplicación Web.”

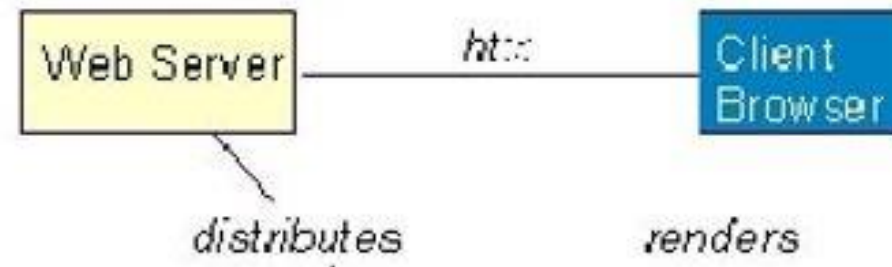


# Extensiones de UML para aplicaciones Web.

## Arquitectura web

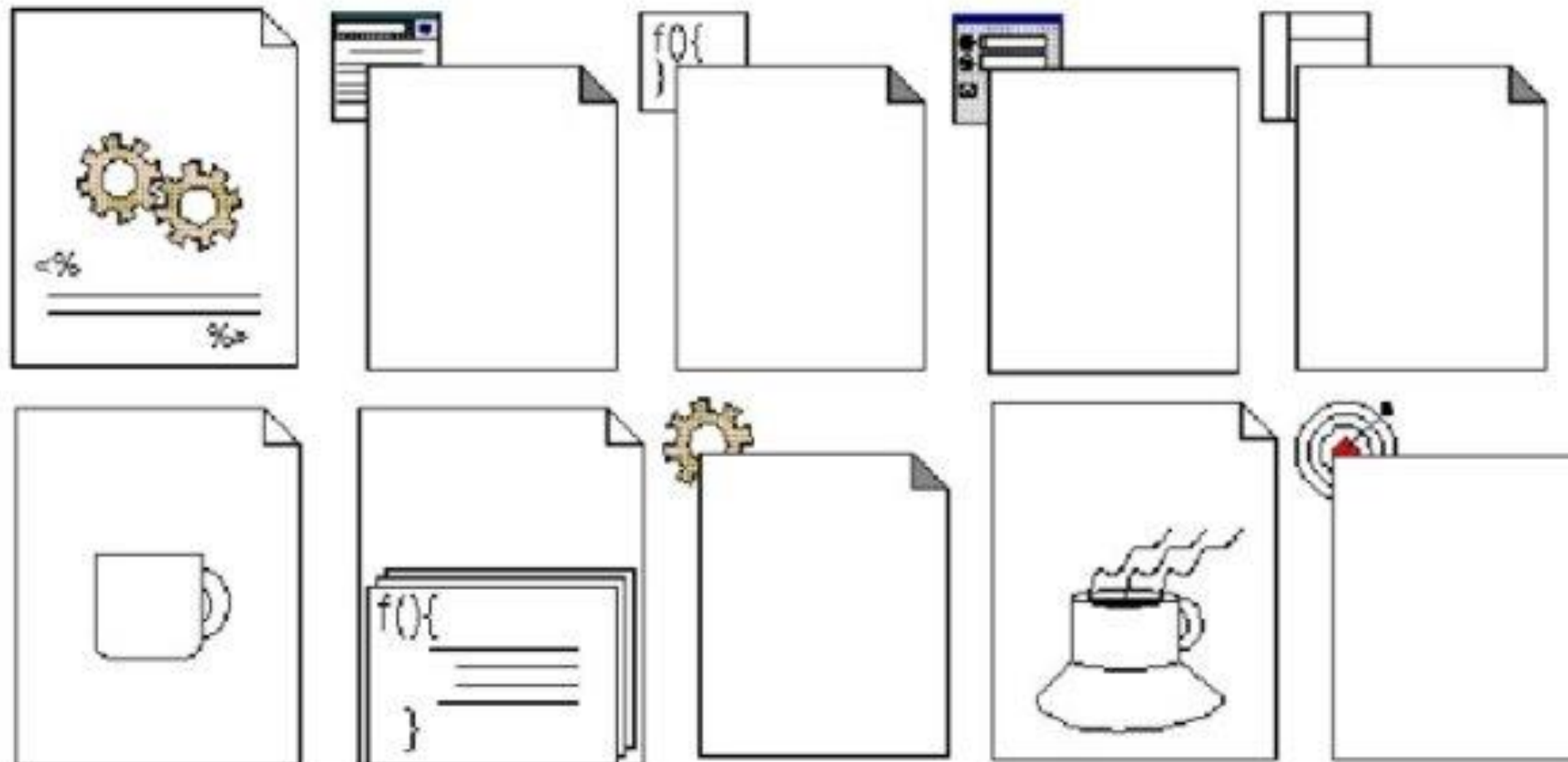


La arquitectura de un sitio Web tiene tres componentes principales: un servidor Web, una conexión de red, y uno o más clientes (browsers). El servidor Web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red, y para ello se usa el protocolo HTTP. Arquitectura Web



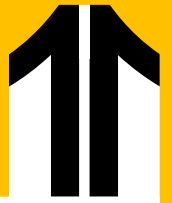
# Extensiones de UML para aplicaciones Web.

## Modelando aplicaciones web


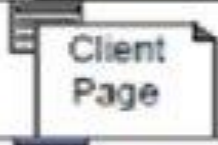
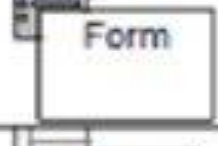




# Extensiones de UML para aplicaciones Web.

## Modelando aplicaciones web

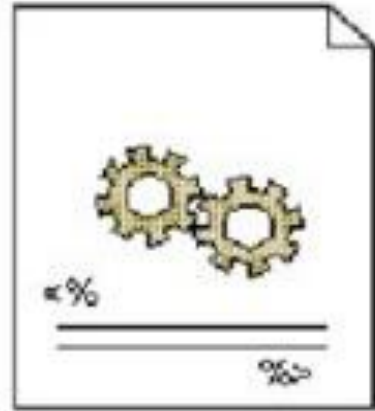


Conallen propone una extensión al UML para diseñar aplicaciones Web:

Estereotipos para las Relaciones entre las Clases		
Nombre	Estereotipo	Descripción
Server Page		Representa una página web que tiene scripts que son ejecutados por el Servidor.
Client Page		Es una página web formateada en HTML
Form		Es una colección de datos de entrada que forman parte de una Client page
Frame Set		Es un contenedor de múltiples páginas web
Target		Son compartimientos de una ventana en el browser donde las páginas web son desplegadas.

# Extensiones de UML para aplicaciones Web.

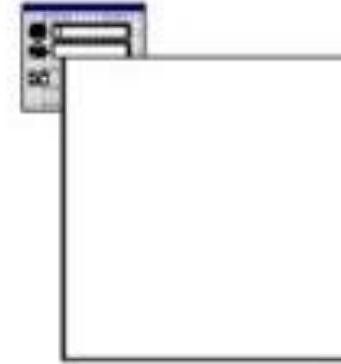
## Modelando aplicaciones web



Los scripts de las páginas del servidor representan los métodos de esta clase. Las páginas del cliente tienen métodos que se ejecutan solamente del lado del cliente, como por ejemplo, Java Applets y controles ActiveX.



### Formularios



Un *formulario* (form) es una colección de campos de entrada: textbox, text area, checkbox, radio button group, button y selection list.

Cuando un formulario es llenado, se envía al servidor usando una operación *submit* solicitada por el usuario típicamente al hacer click en un botón.

# Extensiones de UML para aplicaciones Web.

## Modelando aplicaciones web



ESTEREOTIPOS PARA LAS RELACIONES ENTRE LAS CLASES	
<<Link>>	Representa un apuntador desde una "client page" hacia una "client page" o "server page". Corresponde directamente con un etiqueta <a> (ancla) de HTML.
<<Submit>>	Esta relación siempre se da entre un "form" y una "server page", por supuesto, la "server page" procesa los datos de la "form" le envía "submits"
<<Build>>	Sirve para identificar cuáles "server page" son responsables de la creación de una "client page". Una "server page" puede crear varias "client page", pero una "client page" solo puede ser creada por una sola "server page". Esta relación siempre es unidireccional.
<<Redirect>>	Esta es también una relación unidireccional que indica que una página web redirige hacia otra. En caso de que la página origen sea una "client page" esta asociación corresponderá con la "META" etiqueta y valor HTTP-EQUIV de "Refresh".
<<Forward>>	Es una asociación unidireccional entre una "server page" y otra "server page" o una "client page". Esta asociación representa la delegación de procesamiento de la solicitud de un cliente para un recurso a otra página del lado del servidor.
<<Object>>	Es una relación entre una "client page" y una clase, normalmente uno que representa a un applet, el control ActiveX u otro componente.
<<Include>>	Se da entre una "server page" y otra similar o "client page". Durante la ejecución de la página esta asociación indica que la página incluida es procesada.

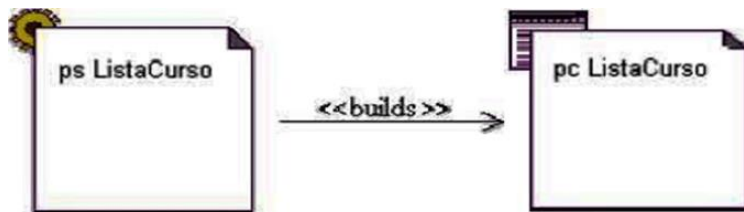


# Extensiones de UML para aplicaciones Web.

## Modelando aplicaciones web

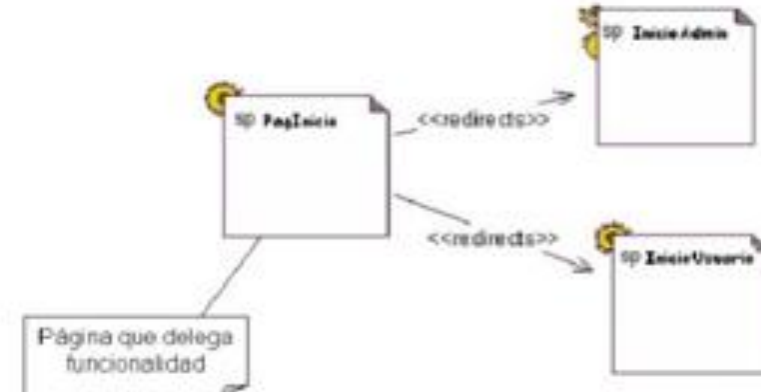


Hay una relación fundamental entre las páginas del servidor y las páginas del cliente, y es que las páginas del servidor crean las páginas del cliente. Esta relación es en una sola dirección, y para modelarla se usa el estereotipo `<<builds>>`. De este modo, se indica cuál página del servidor es encargada de crear la página del cliente. Por ejemplo:



Las páginas del servidor construyen las páginas del cliente

Algunas páginas del servidor podrían redireccionar ciertas solicitudes de procesamiento a otras páginas servidoras (una especie de IF). Permitir modelar estas situaciones es útil para la reutilización. Para esto se utiliza el estereotipo `<<redirects>>`. Por ejemplo:

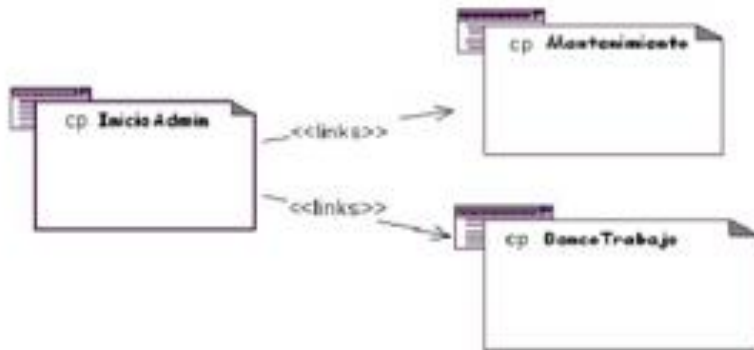


# Extensiones de UML para aplicaciones Web.

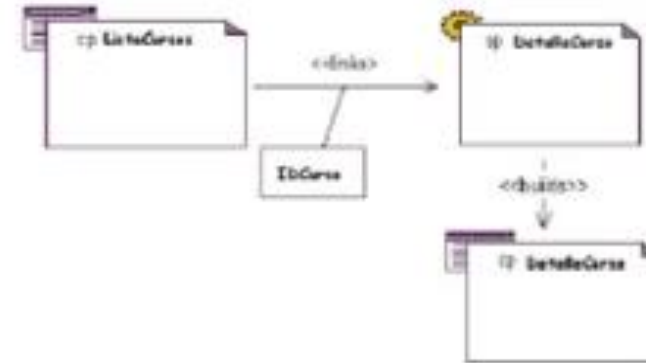
## Modelando aplicaciones web



Otra relación importante en el diseño de aplicaciones Web es el vínculo (*link*, o *anchor*) entre páginas. Las páginas vinculadas podrían ser páginas de cliente o del servidor. El estereotipo `<<links>>` define relaciones entre páginas cliente y otras páginas (cliente o servidoras). Ejemplo:



Si un vínculo (*hyper link*) incluye parámetros, éstos son modelados como atributos del link fuera de la asociación. Por ejemplo:

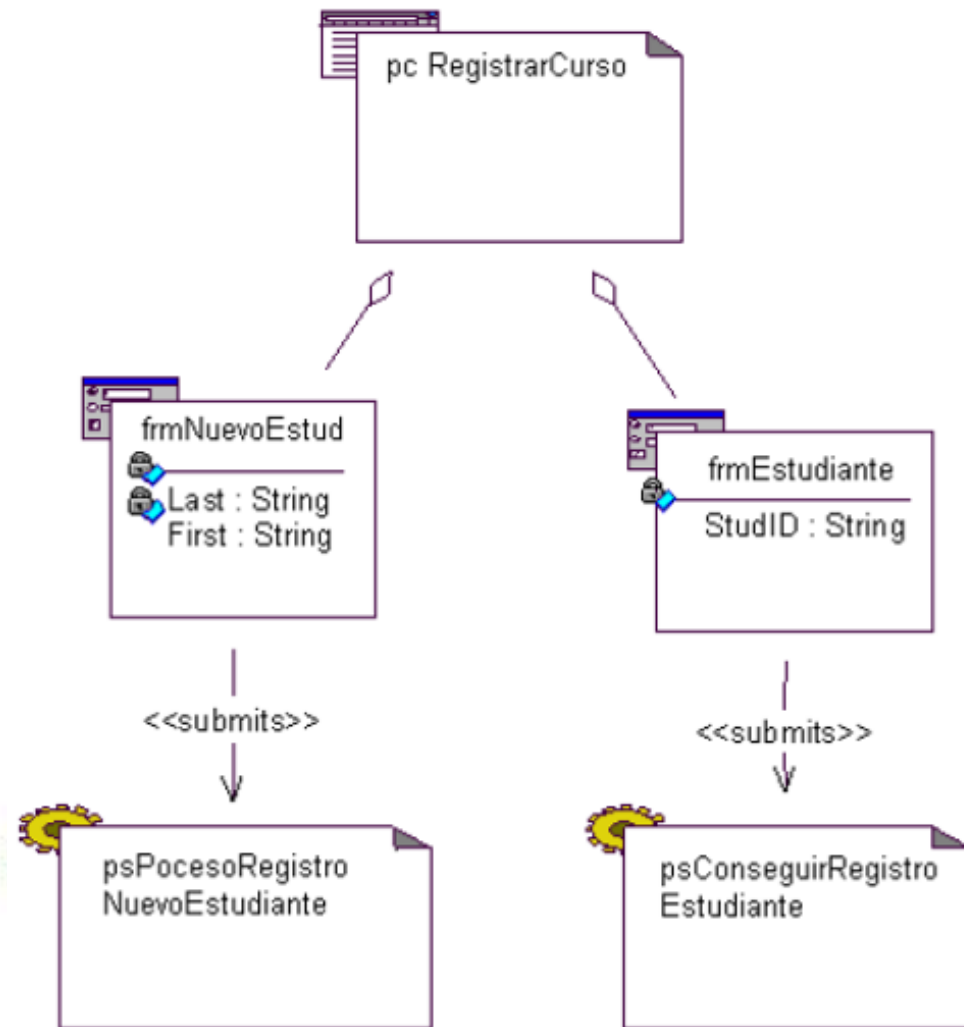


# Extensiones de UML para aplicaciones Web.

## Modelando aplicaciones web



Dado que una página podría tener varios formularios (forms) es posible que desde esta página se acceda a diferentes páginas. Los formularios se modelan con el estereotipo `<<form>>` (un estereotipo por cada formulario). Las páginas cliente contienen formularios. Ejemplo:



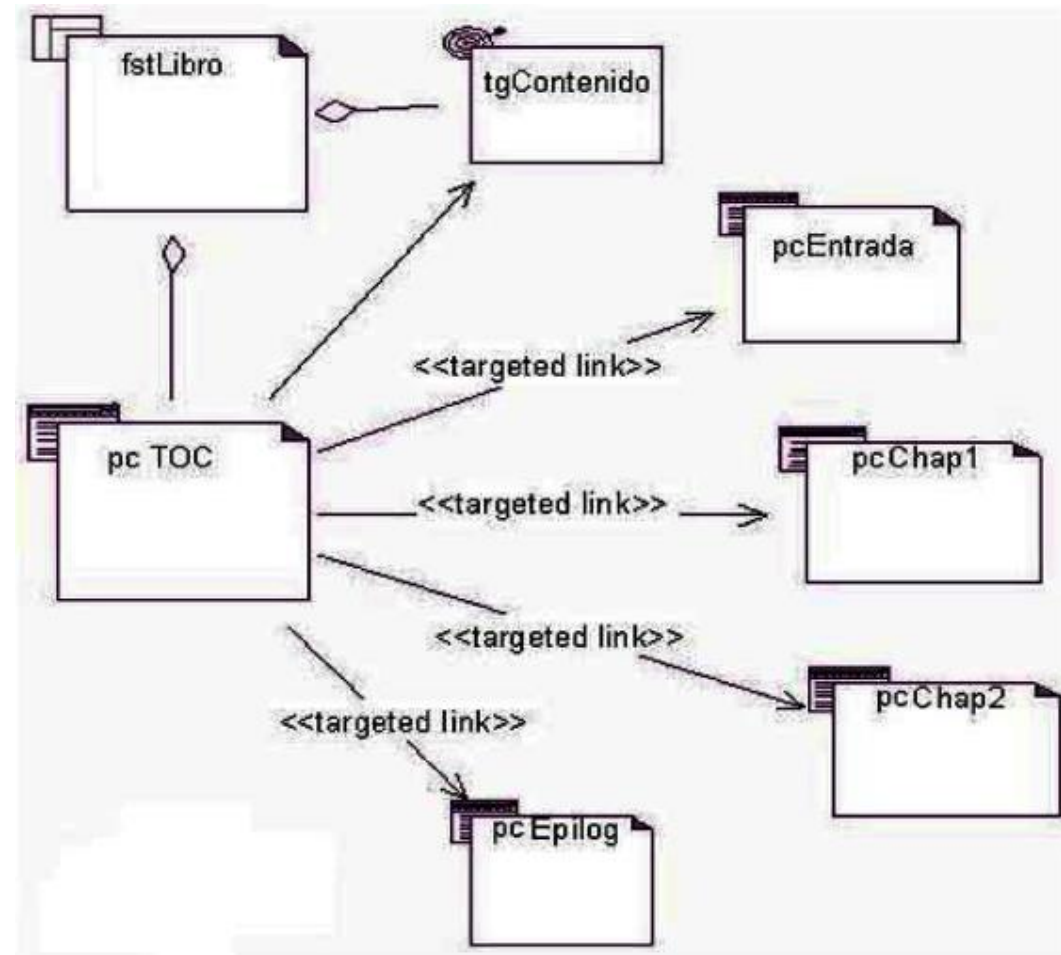
Los formularios hacen un submit a la página del servidor

# Extensiones de UML para aplicaciones Web.

## Modelando aplicaciones web



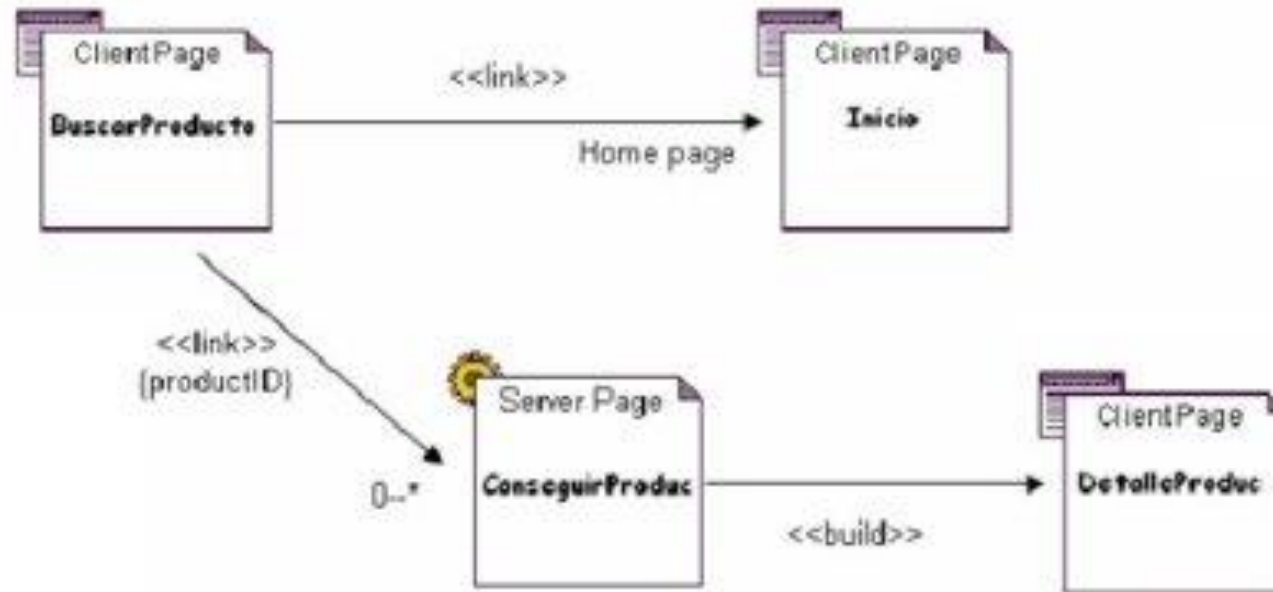
Usando *frames*, una página cliente podría estar compuesta por múltiples páginas al mismo tiempo. Los frames se implementan en HTML usando un *frameset*. Un frameset podría a su vez estar contenido en otro frameset. Las páginas Web contenidas en un frame se llaman *targets*. El estereotipo `<<targeted link>>` hace referencia a páginas que van ser cargadas en un frame distinto del que contiene la página que tiene el link.



Usando framesets y targets.

# Extensiones de UML para aplicaciones Web.

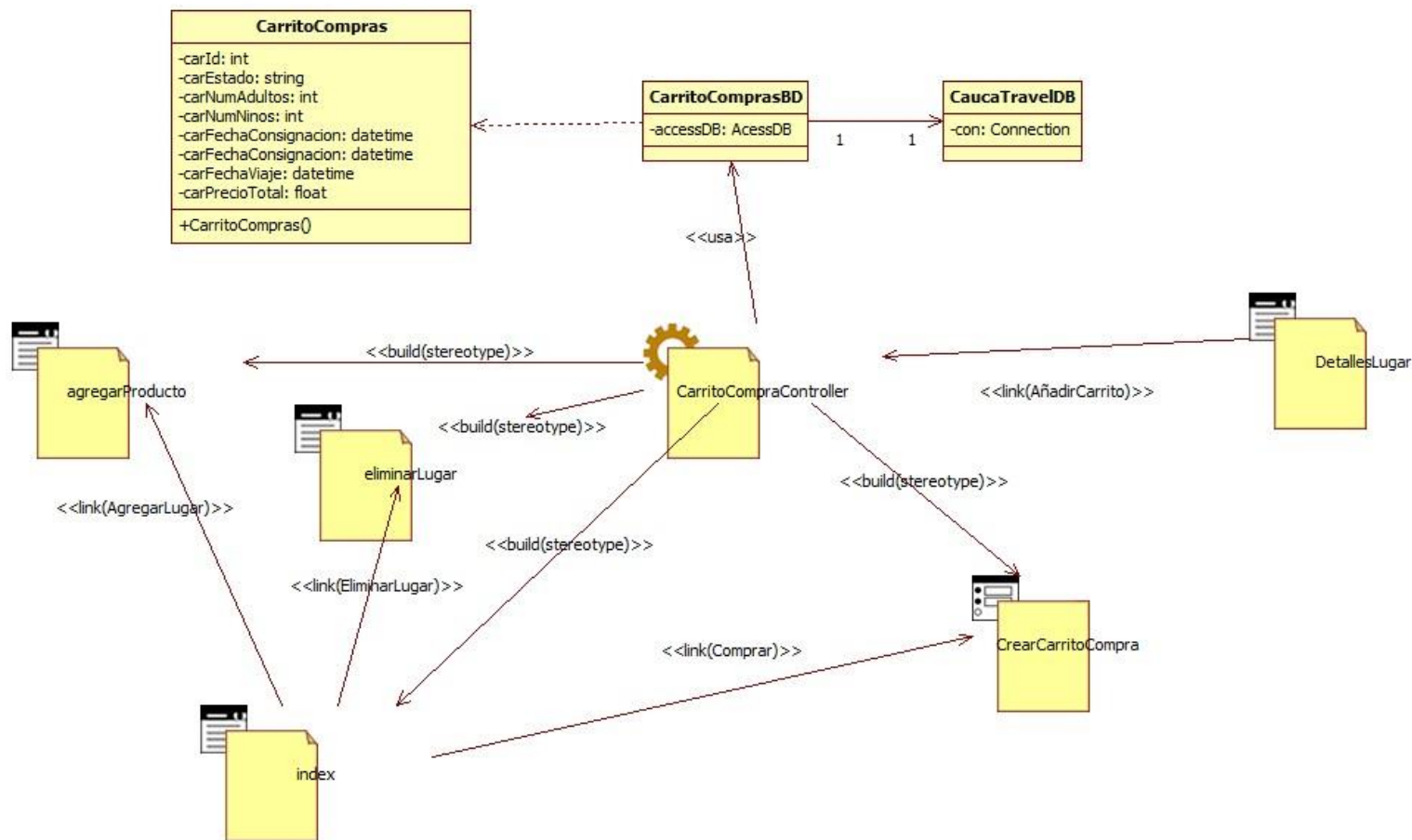
## Modelando aplicaciones web



La página `SearchResults` contiene un número variable de links (0..\*) hacia la página del servidor `GetProduct`. Para cada `productID` se construye una página `ProductDetail` diferente.

# Extensiones de UML para aplicaciones Web.

## Ejemplo







# Modelo de Diseño

**Elaboración del documento de  
Arquitectura de Software**

# Elaboración del Documento de Arquitectura de Software

## Plantilla de Ejemplo



### Reporte de Diseño de Software (RDS)

[Nombre de Gerencia Sponsor del proyecto]

[Nombre del proyecto]

[Mes y Año Inicio del Proyecto]

*[Este documento es la plantilla base para elaborar el documento Reporte de Diseño de Software. Los textos que aparecen entre corchetes son explicaciones de qué debe contener cada sección. Dichos textos se deben seleccionar y sustituir por el contenido que corresponda. En caso que alguna de las secciones del presente documento no aplique a su proyecto pueden usarse las frases "No hay cambios", "No hay impacto en esta sección", "La solución que se está implementando no tiene impacto en esta sección", "No aplican para el proyecto" (No borrar secciones del documento)]*

Elaborado por:	Revisado por:	Aprobado por:
Fecha: / /	Fecha: / /	Fecha: / /

### Contenido

1.	Introducción.....	2
1.1.	Propósito .....	2
1.2.	Alcance.....	2
1.3.	Definiciones, Acrónimos y Abreviaturas .....	2
1.3.1.	Definiciones .....	2
1.3.2.	Acrónimos .....	2
1.3.3.	Abreviaturas .....	2
1.4.	Referencias.....	3
2.	Vista General de la Arquitectura .....	3
3.	Metas y Restricciones de la Arquitectura .....	4
4.	Vista de Casos de Uso .....	4
5.	Vista Lógica.....	4
5.1.	Realización de Casos de Uso – Modelo de Análisis.....	5
5.1.1.	Código del CUS – Nombre del CUS .....	5
5.1.2.	Diagrama de Clases de Análisis.....	5
5.2.	Modelo Conceptual .....	5
5.3.	Modelo Lógico.....	5
5.4.	Modelo de Diseño.....	6
5.4.1.	Vista de Capas y Subsistemas .....	6
5.4.1.1.	Capa de Presentación.....	6
5.4.1.2.	Capa de Negocio.....	6
5.4.1.3.	Capa de Integración.....	6
5.4.1.4.	Capa de Datos.....	6
5.4.1.5.	Capa de Entidad .....	6
5.4.1.6.	Capa de Interfaces o Elementos Comunes .....	6
5.4.2.	Realización de Casos de Uso – Modelo de Diseño.....	6
5.4.2.1.	Código del CUS – Nombre del CUS.....	6
5.4.2.2.	Diagrama de Clases de Diseño .....	6
6.	Vista de Procesos .....	7
7.	Vista de Despliegue.....	7
8.	Vista de Implementación .....	7
9.	Vista de Integración del Software .....	7
9.1.	Criterios de Integración de Software .....	9
9.2.	Secuencia de Integración .....	9
9.3.	Entorno Necesario para la Integración .....	10
10.	Vista de Datos.....	11
11.	Tamaño y Desempeño .....	11



# ACTIVIDAD DE APRENDIZAJE

En Equipo



EN EQUIPOS DE 3 O 4 REALIZAR UN DOCUMENTO DE ARQUITECTURA



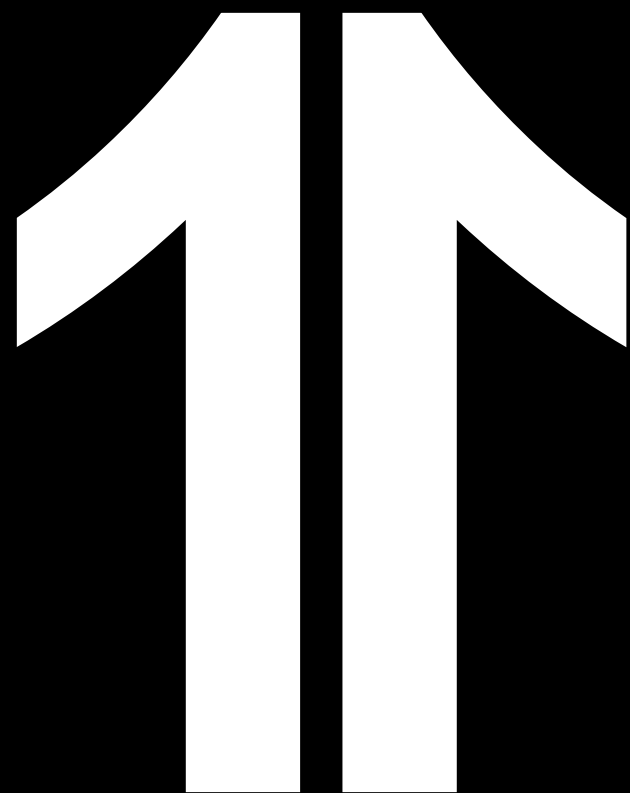
# CONCLUSIONES



- El diagrama de clases de diseño, es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.
- El diagrama de secuencia describe la dinámica del sistema, describiendo las interacciones entre un grupo de objetos mostrando de forma secuencial los envíos de mensajes entre objetos.
- El modelo de persistencia o modelo físico es un modelo de datos de bajo nivel. Proporciona conceptos que describen los detalles de cómo se almacenan los datos en el ordenador.



# EVALUACIÓN T3



**UPN**

**UNIVERSIDAD  
PRIVADA  
DEL NORTE**