**DISEÑO ORIENTADO A OBJETOS**

**TEMARIO**

Tema 1: Capas lógicas de la arquitectura

1. Definición de capas
2. Creación de subsistemas
3. Creación de componentes.

Tema 2: Diseño de casos de uso con patrón arquitectónico MVC

1. Realizaciones de diseño de casos de uso
2. Diagrama de Clases
3. Diagrama de Secuencia del flujo básico y subflujos de un mantenimiento.

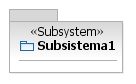
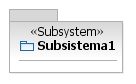
# 1. CAPAS LÓGICAS DE LA ARQUITECTURA

Antes de indicar cómo se realiza las realizaciones de diseño de un caso de uso, primero se mostrará la organización de los componentes (clases de diseño e interfaces) en capas, subsistemas y librerías que utilizaremos en el curso, aplicando patrón arquitectónico MVC:

|  |  |  |
| --- | --- | --- |
| **Capa** | **Subsistema/Librerías** | **Componentes** |
|  |  | Clases estereotipadas:   * Páginas HTML: <<Client Page>> y <<HTML Form>> * Páginas JSP: <<Server Page>>, <<Client Page>> y   <<HTML Form>> |
|  |  | Clase estereotipada para servlets: <<Http Servlet>> |
|  |  | Clases de diseño: *beans*. |
|  | Clases de diseño: clases utilitarias. |

### Tabla 3.1. Capas, subsistemas, librerías y elementos de diseño según patrón arquitectónico MVC.

En la siguiente tabla, se muestra la organización de las clases de diseño e interfaces en capas, subsistemas y librerías que utilizaremos en el curso, aplicando patrón arquitectónico MVC y patrón de diseño DAO:



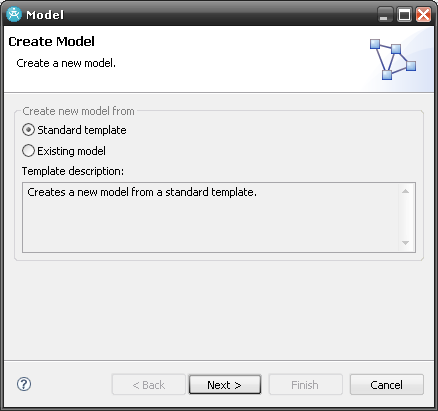
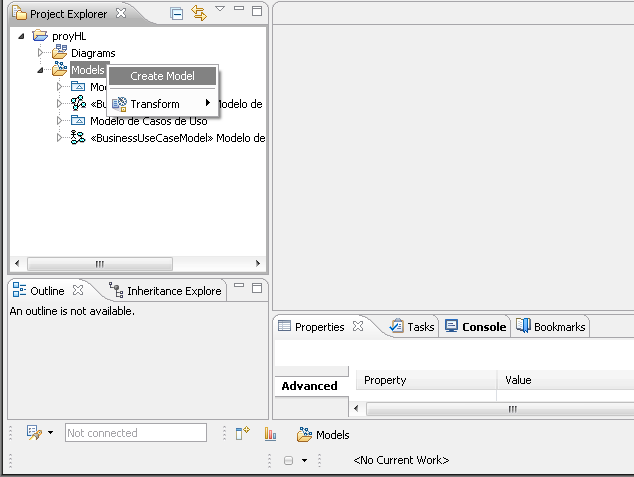
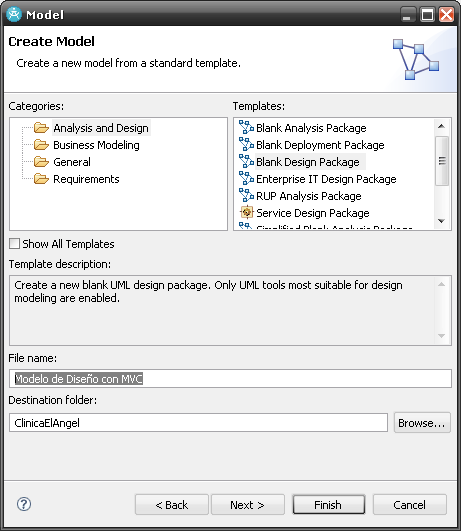
|  |  |  |
| --- | --- | --- |
| **Capa** | **Subsistema/Librerías** | **Componentes** |
|  |  | Clases estereotipadas:   * Páginas HTML: <<Client Page>> y <<HTML Form>> * Páginas JSP: <<Server Page>>, <<Client Page>> y   <<HTML Form>> |
|  |  | Clase estereotipada para servlets: <<Http Servlet>> |
|  |  | * Clases de diseño: servicios, *beans* y clases DAO. * Interfaces que presentan las operaciones de acceso a una tabla. |
|  | Clases de diseño: clase abstracta *DAOFactory* y sus clases hijas. |
|  | Clases de diseño: clases utilitarias. |

### Tabla 3.2. Capas, subsistemas, librerías y elementos de diseño según patrón arquitectónico MVC y patrón DAO.



**Definición de capas, subsistemas y componentes de diseño según MVC**

1. En la vista de explorador de proyectos, crear el Modelo de Diseño.



**1**

**2**

**3**

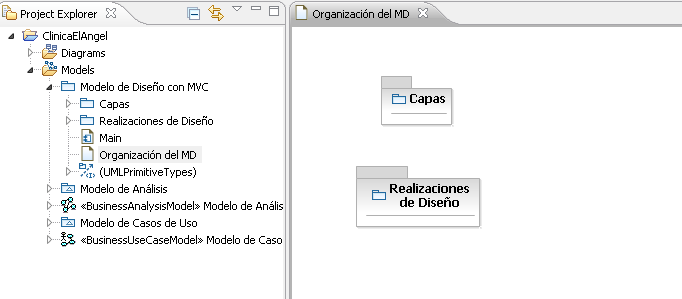
**4**

**5**

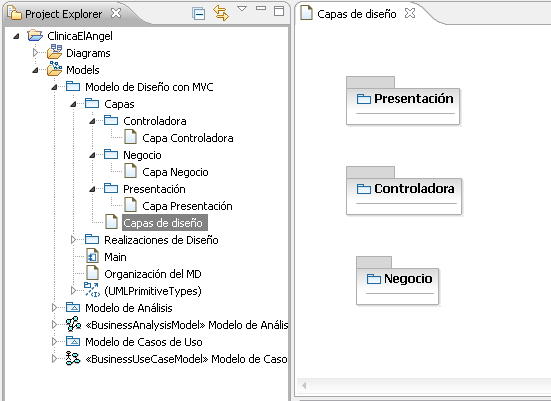
**6**

**Next** hasta ir a la ventana de capacidades, donde activará las opciones de UML

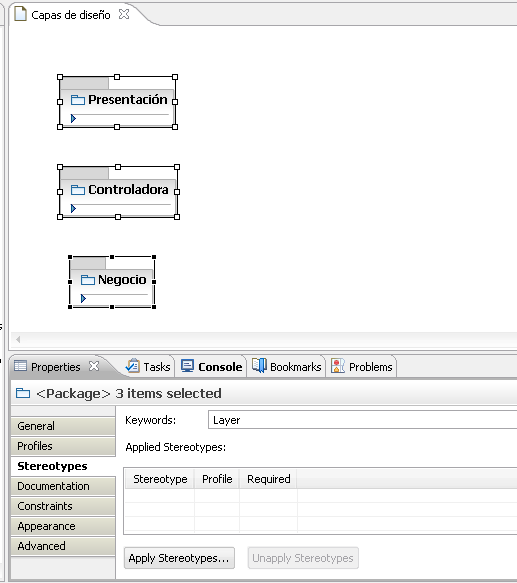
1. Cree un diagrama de formato libre de nombre “Organización del MD” y agregue dos paquetes: Capas y Realizaciones de Diseño.



1. En Capas renombre el diagrama de formato libre a “Capas de diseño” y agregue tres paquetes: Presentación, Controladora y Negocio. Luego, renombre los diagramas de cada paquete, así:



1. Cambie el estereotipo de las capas a *Layer* y relaciona las capas con dependencia.



**1**

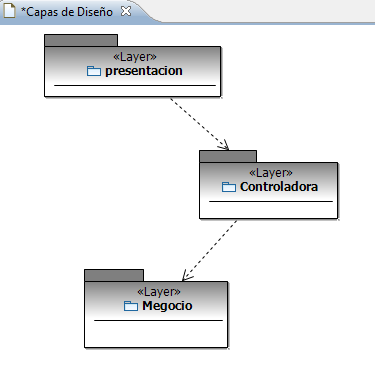
Seleccione las tres capas con **Ctrl + A**

**3**

Edite ***Layer***

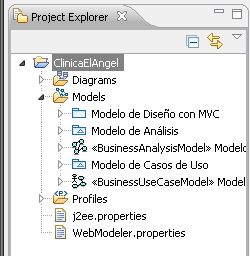
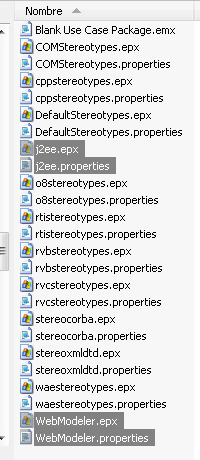
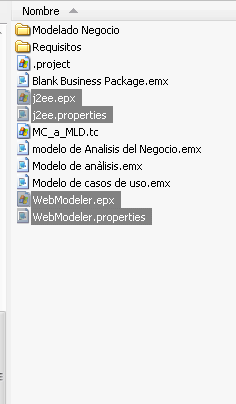
**2**

Se habrá cambiado el estereotipo de las capas:



**5.3.**

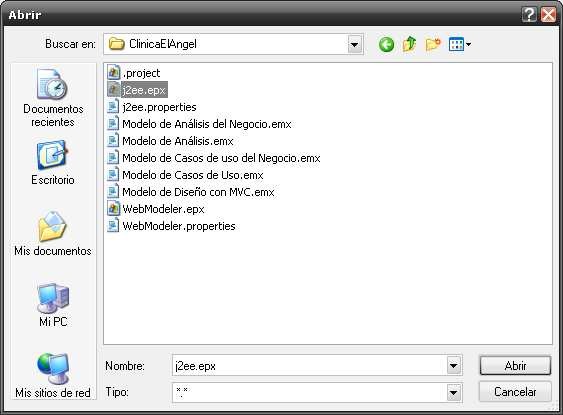
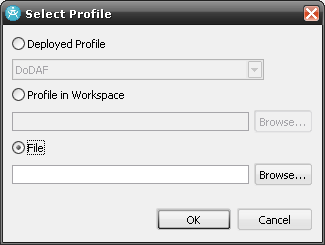
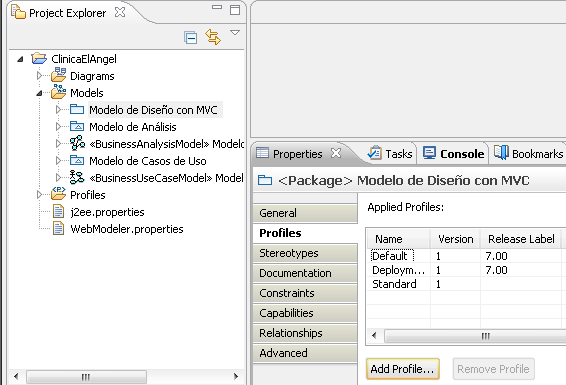
1. Antes de empezar a crear los elementos de diseño asigne los perfiles correspondientes.
   1. Desde el explorador de windows, copie los perfiles **j2ee** y **WebModeler** de la carpeta *profiles* a la carpeta del proyecto:
   2. A continuación, desde el explorador de proyectos en el RSA, seleccione el proyecto y presione **F5** para cargar los perfiles. Debe quedar así:



**1**

Seleccione el proyecto y luego presione **F5.**

Ahora, especifique los perfiles para el modelo de diseño así:



1

2

3

4

5

8 6

9

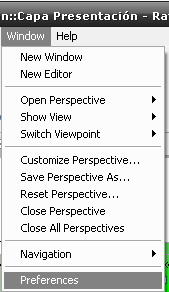
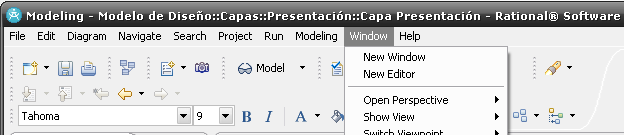
La misma operación

se realiza para 7

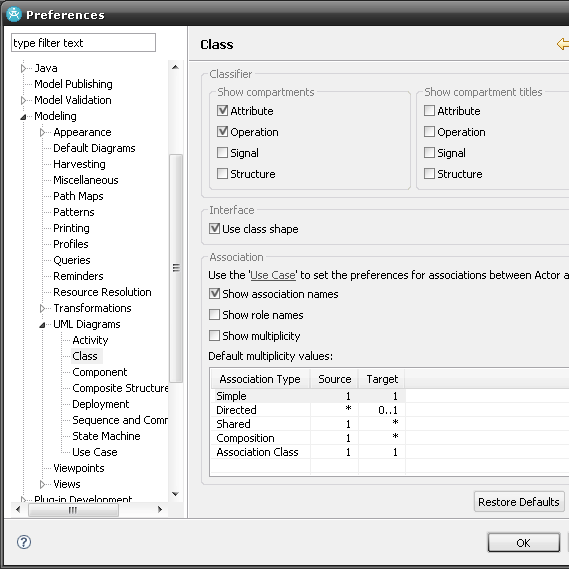
agregar el perfil

# WebModeler.

1. Ahora, para crear las clases de diseño configuremos los diagramas de clases desde **Window >> Preferences**:



**1**



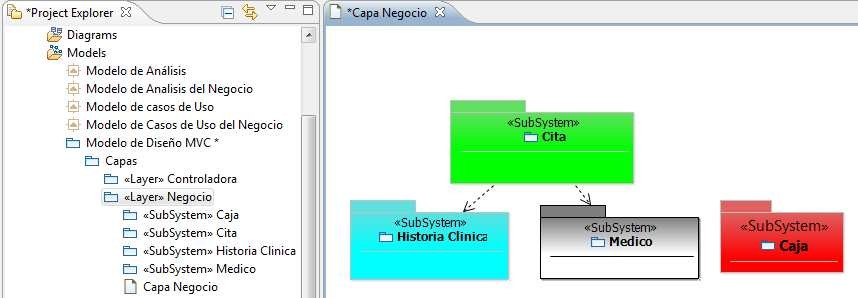
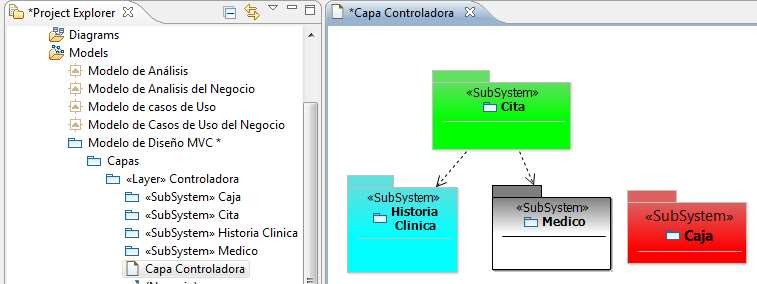
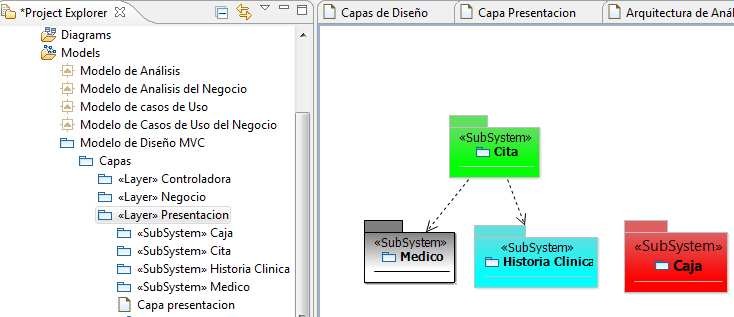
**2**

**3**

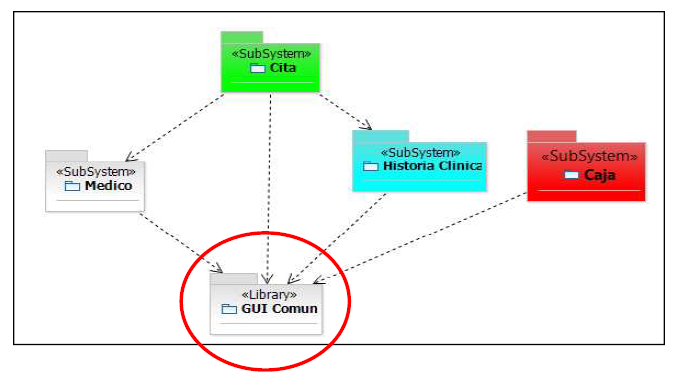
Desactive las

opciones indicadas.

6. A continuación, en cada capa agregue los mismos paquetes que creó en la arquitectura de análisis manteniendo los mismos colores. Luego, cambie sus estereotipos a ***Subsystem***. En la figura se muestra los subsistemas que contiene los casos de uso que hemos trabajado hasta el momento.



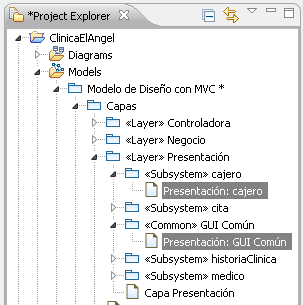
1. Ahora agregue otros paquetes que agruparán clases de uso común o clases utilitarias y, luego, asígneles el estereotipo correspondiente, según sea el caso: ***Common*** *o* ***Library.*** En este caso, solo se agregarán otros paquetes en las capas: Presentación y Negocio. En el paquete presentación agregue el paquete (estereotipo) **GUI Comun**



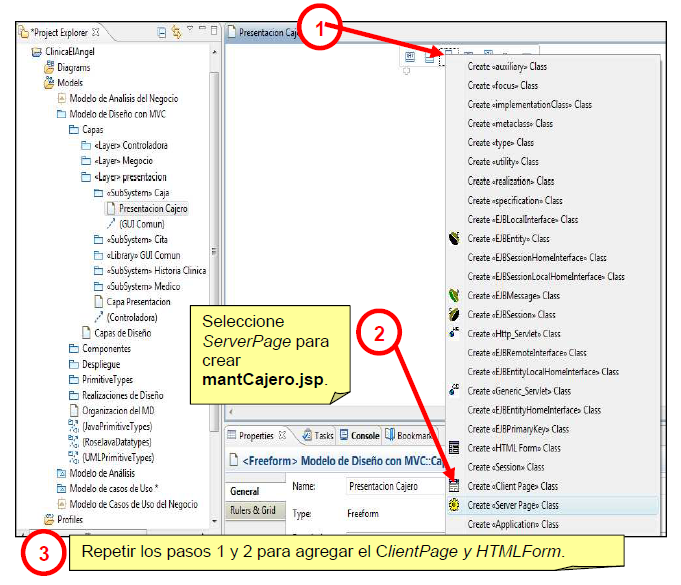
En la capa de negocio, agregar los paquetes AdmDAO y Util y asignarles el estereotipo **library**. El cambio debe mostrarse así:



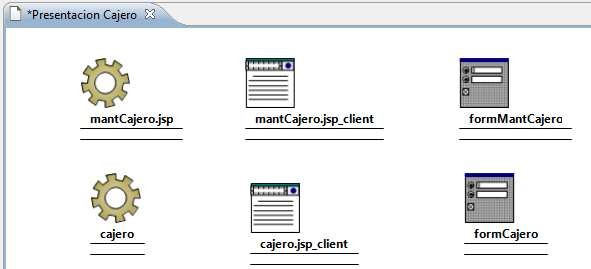
1. A continuación, empezaremos a trabajar con la **Capa Presentación** para realizar el CU Mantener Cajero.
   1. Cambie el nombre de los diagramas *Main* de los paquetes:



* 1. Luego, crearemos los JSP del CU Mantener Cajero los cuales son **mantCajero.jsp** y **cajero.jsp**. Por cada JSP que tenga formularios, se crea tres clases estereotipadas, tal como se muestra a continuación:

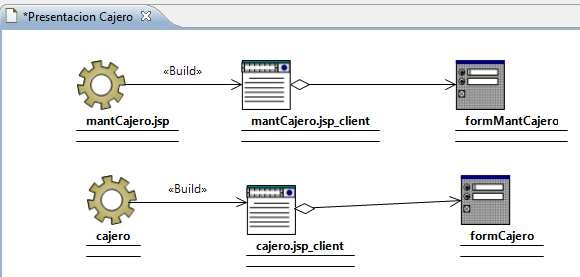


Debe quedar así:



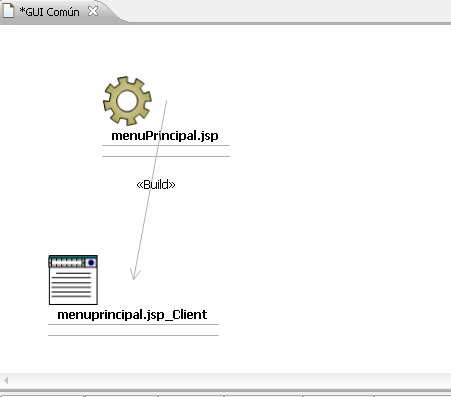
* 1. Ahora agregue las relaciones entre las clases. Para la asociación entre

<<Server Page>> y <<Client Page>> indique el estereotipo ***Build*** tal como se hizo con las capas y para <<Client Page> a <<HTML Form>> agregue una agregación por composición directa.

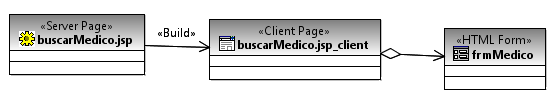
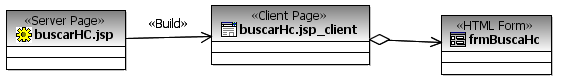


* 1. Ahora cree **menuPrincipal.jsp** en GUI Común

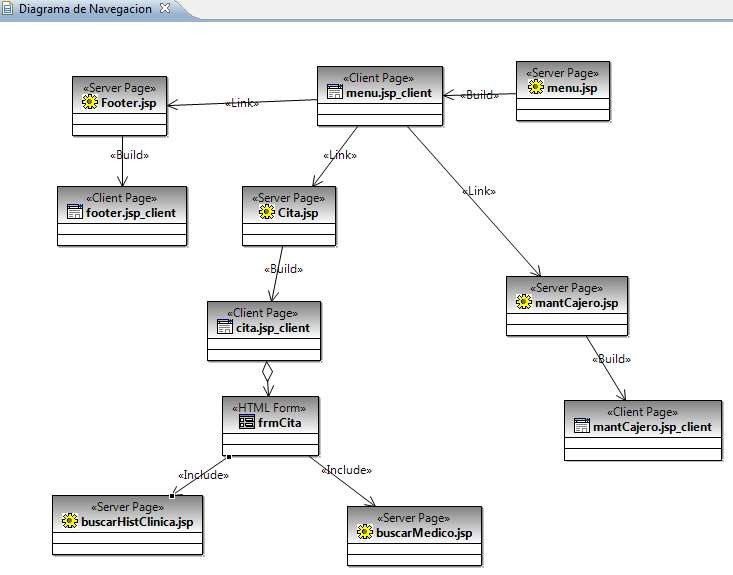
NOTA: Como esta página no contiene formularios no se le creará un *HTML Form.*



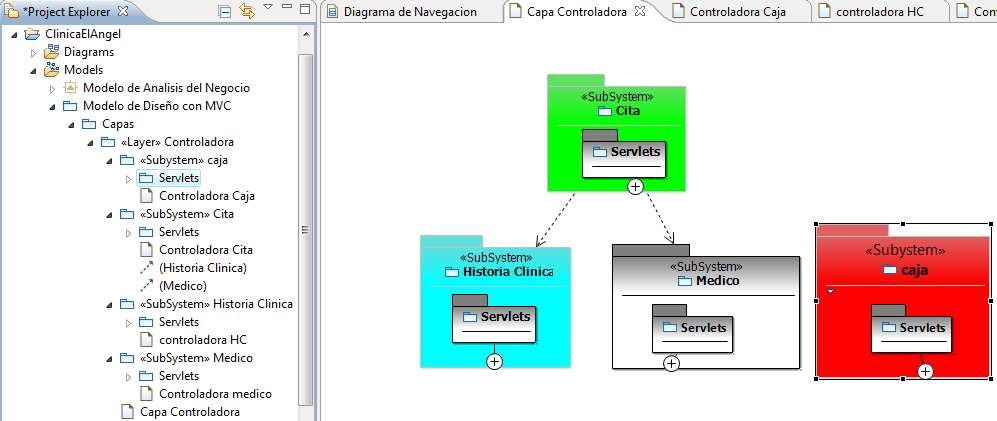
* 1. Lo mismo se trabajará para los paquetes Historia Clínica, Cita y médico.



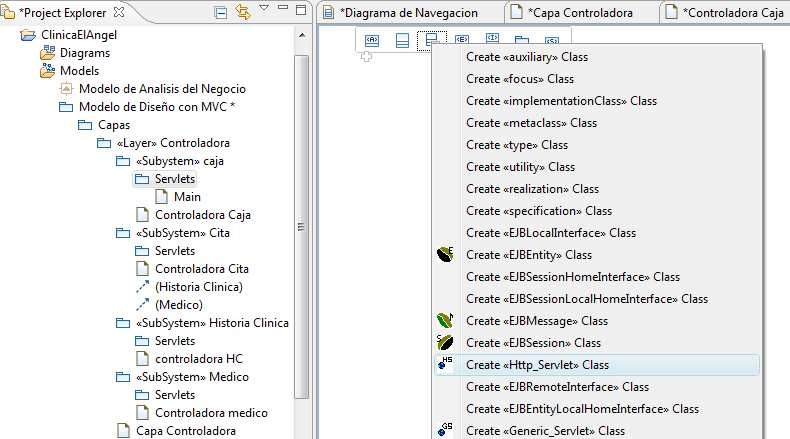
* 1. Ahora, en la capa de Presentación, se agrega un diagrama de clases el cual llamamos Diagrama de Navegación y ahí se coloca la navegación de la capa de presentación



1. Luego, agregue los servlets en la **Capa Controladora** para realizar los CU Mantener Cajero. CU buscar historia Clínica y Generar cita.
   1. Primero, cambie los nombres del diagrama *Main* de cada paquete en el cual trabajará y agregue el paquete **servlets**:



* 1. Ahora crearemos el servlet para este subsistema a partir de una clase estereotipada, tal como se muestra a continuación:



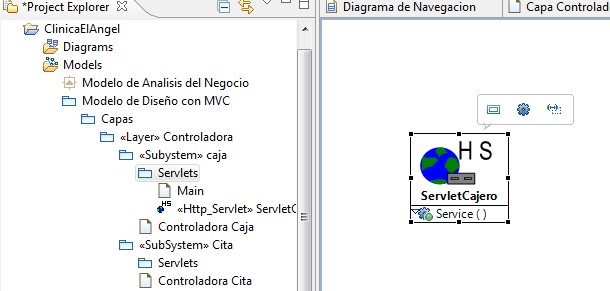
**1**

Seleccione

*Http\_Servlet* para crear **ServletCajero**.

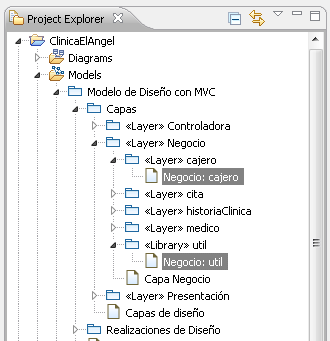
**2**

* 1. Por último, agregue la operación **Service().**

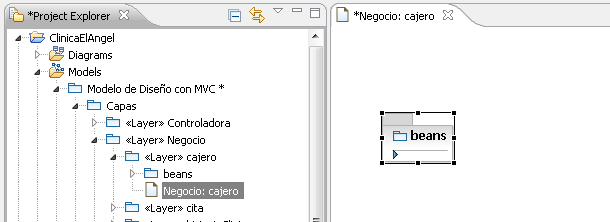


**1**

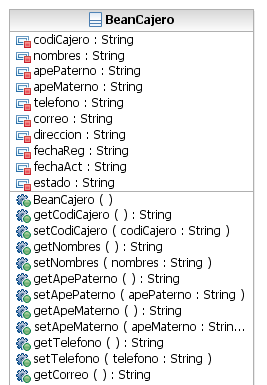
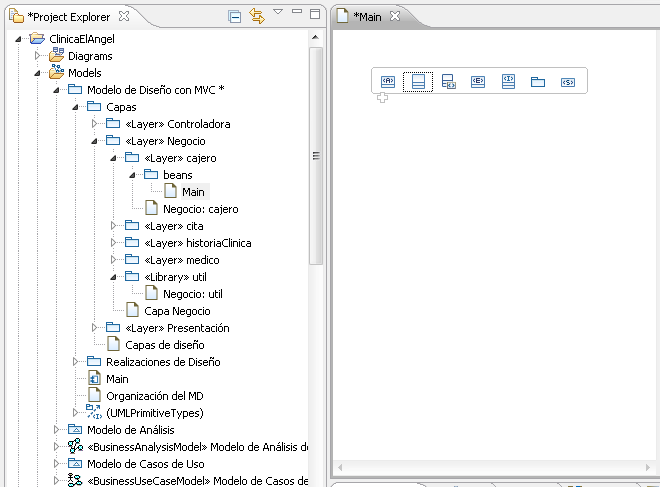
1. Luego, agregue las clases de diseño en la **Capa Negocio** para realizar el CU Mantener Cajero.
   1. Primero, cambie el nombre del diagrama *Main* de los paquetes en los cuales trabajará:



* 1. En el subsistema cajero agregue el paquete **beans**



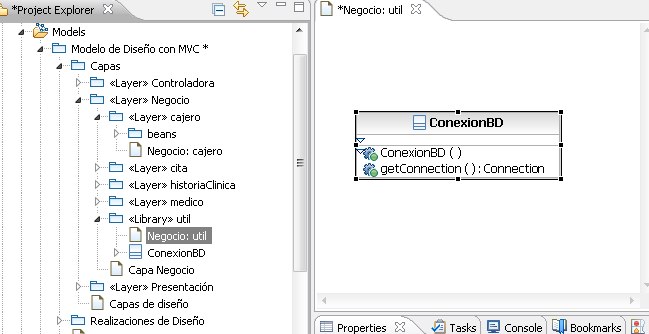
* 1. A continuación, en el paquete beans, agregue el **BeanCajero** con sus atributos y operaciones de acceso (get/set).



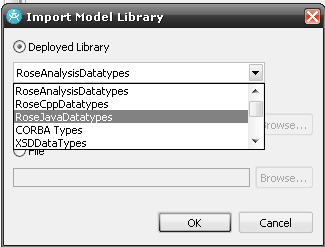
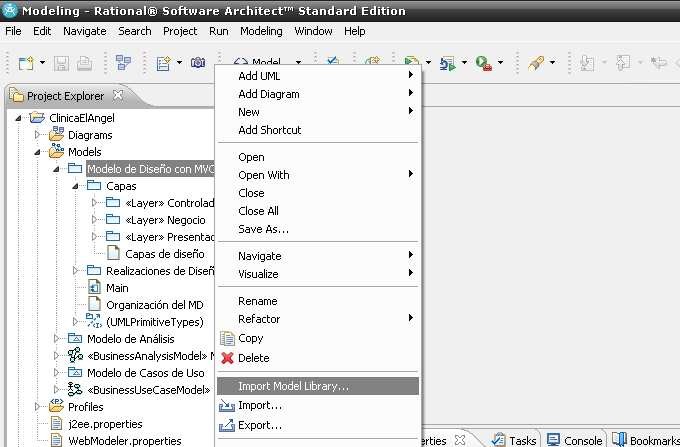
**1**

* 1. Por último, en el paquete util agregue la clase ConectaBD con sus operaciones.

**NOTA: Para mostrar parámetros de operaciones (funciones y procedimientos) seleccionar con botón derecho: Filtros/ Mostrar signatura.**



1. Finalmente, agregue las librerías de JAVA.



**1**

**2**

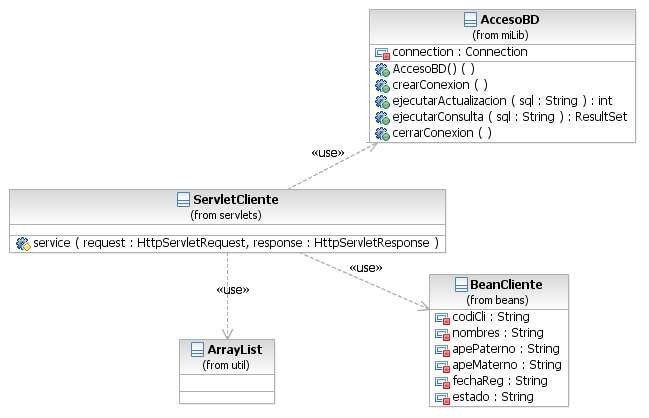
**3**

# DISEÑO DE CASOS DE USO CON MVC

A continuación, se explica la definición de algunos elementos que se utilizarán en las realizaciones de diseño de un caso de uso.

## Diagrama de clases

Un diagrama de clases 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.



## Figura 1. Diagrama de clases de diseño.

En las siguientes tablas se muestran las relaciones que pueden existir entre clases. La descripción de cada una, permitirá entender la estructura de clases diseñada para una funcionalidad que será implementada en JAVA:

|  |  |  |
| --- | --- | --- |
| **Tipo de relación** | **UML** | **Java** |
| **Herencia** |  | **public class** ClaseA {  **IIMás código**  }  **public class** ClaseB **extends** ClaseA {  **IIMás código**  } |
| **Implementación** |  | **public interface** InterfazX {  **IIMás código**  }  **public class** ClaseY **implements** InterfazX {  **IIMás código**  } |
|  |

### Tabla 1. Relación de herencia e implementación.

|  |  |  |
| --- | --- | --- |
| **Tipo de dependencia** | **UML** | **Descripción** |
| **<<use>> (De uso)** |  | El funcionamiento del origen depende del funcionamiento del destino. |

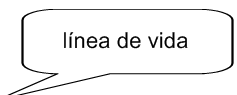
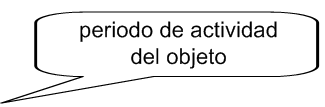
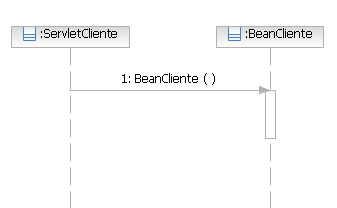
**Tabla 2. Relaciones de dependencia.**

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

**Tabla 2. Relaciones de dependencia. (Continuación)**

* 1. **Diagrama 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.



### Figura 2. Diagrama de Secuencia con mensaje sincrónico.

* 1. **Línea de vida de un objeto**

Dado que representa la dinámica del sistema, el diagrama de secuencia hace entrar en acción las instancias de clases que intervienen en la realización de la subfunción a la que está vinculado. A cada instancia se asocia una línea de vida que muestra las acciones y reacciones de la misma, así como los periodos durante los cuales ésta está activa, es decir, durante los que ejecuta uno de sus métodos.

## Mensajes

Para interactuar entre sí, los objetos se envían mensajes. Durante la recepción de un mensaje, los objetos se vuelven activos y ejecutan el método del mismo nombre. Un envío de mensaje es, por tanto, una llamada a un método y se representan mediante flechas horizontales que unen la línea de vida del objeto emisor con la línea de vida del objeto destinatario. Existen diferentes tipos de mensajes:

El **mensaje sincrónico** es el utilizado con mayor frecuencia. Su uso significa que el expedidor del mensaje espera que la activación del método mencionado por el destinatario finalice antes de continuar su actividad.

En los **mensajes asincrónicos**, el expedidor no espera el término de la activación invocada por el destinatario. Esto se produce al modelar sistemas en los que los objetos pueden funcionar en paralelo (es el caso de los sistemas multi- thread, donde los tratamientos se efectúan en paralelo).

## E) Fragmentos combinados

Para un diagrama de secuencia que representa procedimientos complejos hay un número de mecanismos que permiten agregar un grado de lógicas de procedimientos a los diagramas y que a la vez vienen bajo el encabezado de fragmentos combinados. Un fragmento combinado es una o más secuencias de procesos incluidas en un marco y ejecutadas bajo circunstancias nombradas específicas. Los fragmentos disponibles son los siguientes:

1. El fragmento **Alternative** (denotado “alt”) modela estructuras if…else.
2. El fragmento **Option** (denotado “opt”) modela estructuras switch.
3. El fragmento **Break** modela una secuencia alternativa de eventos que se procesa en lugar de todo del resto del diagrama.
4. El fragmento **Parallel** (denotado “par”) modela procesos concurrentes.
5. El fragmento de secuenciado **Weak** (denotado “seq”) incluye un número de secuencias para las cuales todos los mensajes se deben procesar en un segmento anterior, antes de que el siguiente segmento pueda comenzar, pero que no impone ningún secuenciado en los mensajes que no comparten una línea de vida.
6. El fragmento de secuenciado **Strict** (denotado “strict”) incluye una serie de mensajes que se deben procesar en el orden proporcionado.
7. El fragmento **Negative** (denotado “neg”) incluye una serie de mensajes inválidos.
8. El fragmento **Critical** incluye una sección crítica.
9. El fragmento Ignore declara un mensaje o mensajes que no son de ningún interés si este aparece en el contexto actual.
10. El fragmento **Consider** es el opuesto del fragmento Ignore: cualquier mensaje que no se incluya en el fragmento Consider se debería ignorar.
11. El fragmento **Assertion** (denotado “assert”) designa que cualquier secuencia que no se muestra como un operando de la aserción es inválida.
12. El fragmento **Loop** incluye una serie de mensajes que están repetidos.

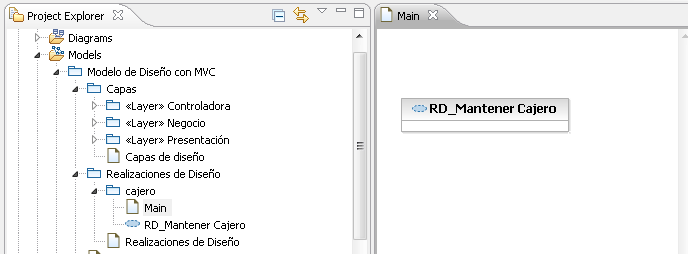
En la siguiente figura se muestra un ejemplo de los fragmentos combinados más utilizados:



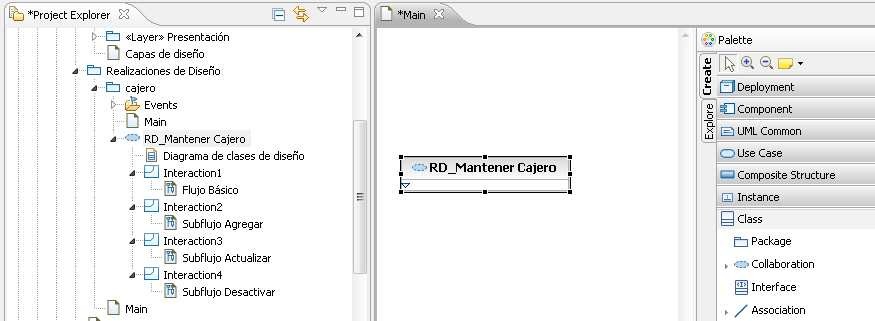
### Figura 3. Fragmentos combinados más utilizados.

**Realizaciones de diseño de un caso de uso**

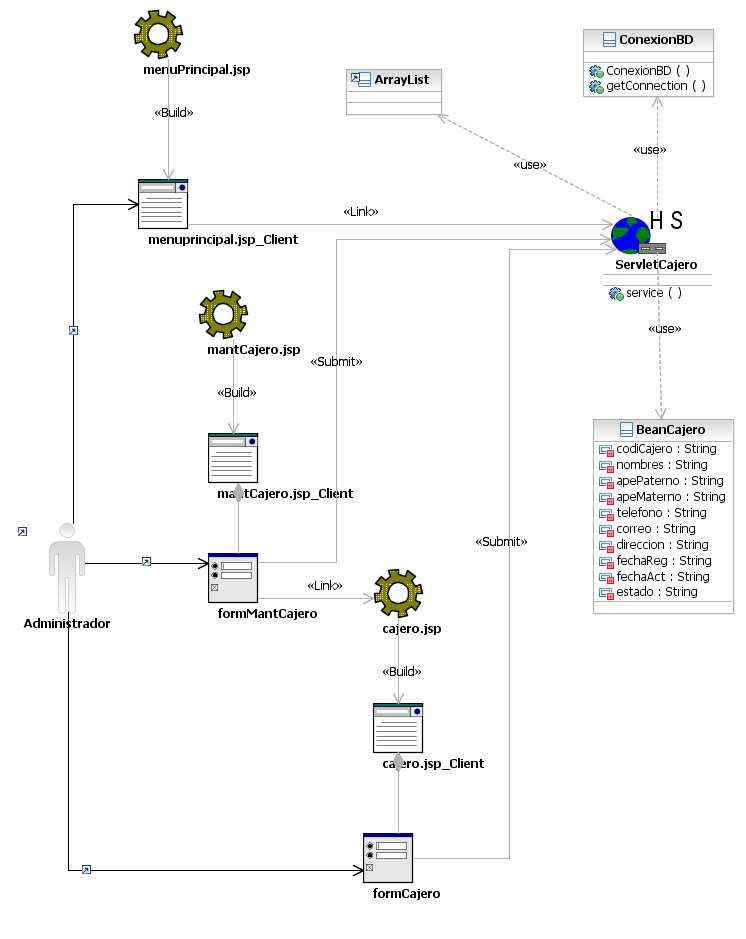
1. En el paquete Realizaciones de Diseño cree el paquete **cajero** con una colaboración para elaborar la Realización de Diseño del CU Mantener Cajero.



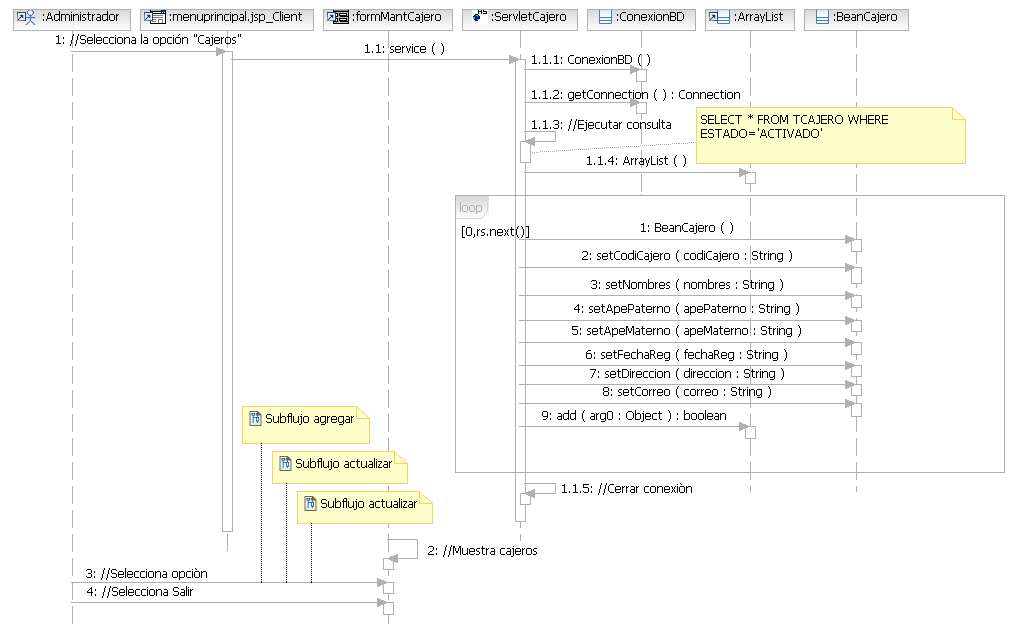
1. Luego, agregue un diagrama de clases y diagramas de secuencia para el flujo básico y subflujos.



1. Ahora realice el Diagrama de Clases de Diseño. Para ello: arrastre los elementos de diseño de las diferentes capas y agregue sus relaciones. Debe quedar así:



1. A continuación, se presenta el Diagramas de Secuencia del flujo básico.

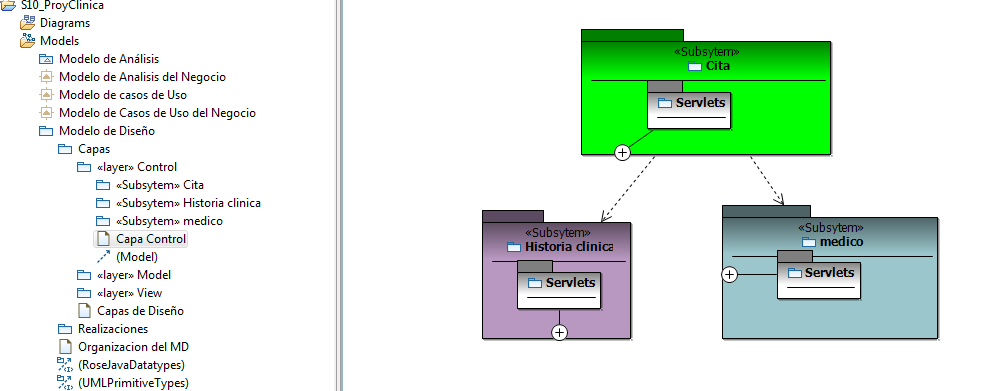


# 3. DISEÑO DE CASOS DE USO CON MVC

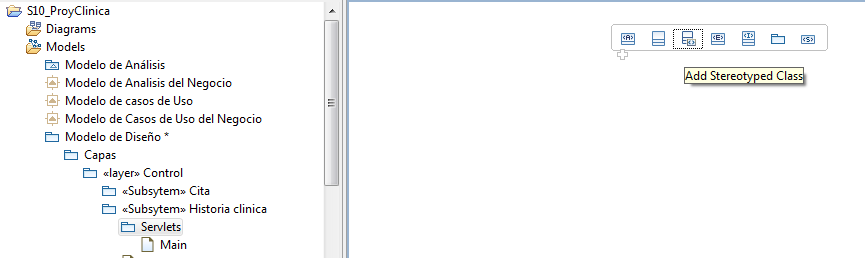
A continuación, se indica los pasos para las realizaciones de diseño de un caso de uso aplicando patrón de diseño DAO, Para el ejemplo se utilizará la ECU del caso buscar Historia clínica y Generar Cita.

**PASO 1**Crear la clase control de la capa control del Subsystem Historia Clinica

Ir a la capa control a el Subsystem Historia Clinica y crear el servlet “SevletHc”

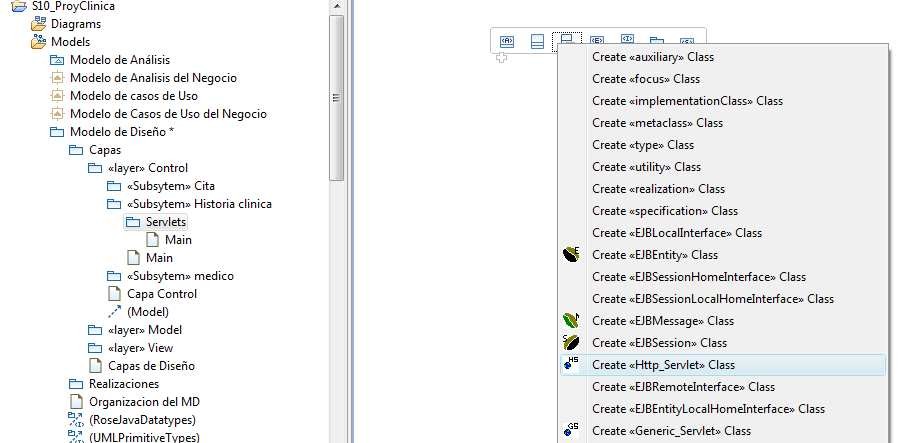


Seleccionar del menú contextual la clase estereotipada



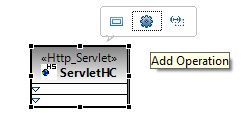
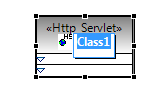
**1**

Seleccionamos la clase http Servlet



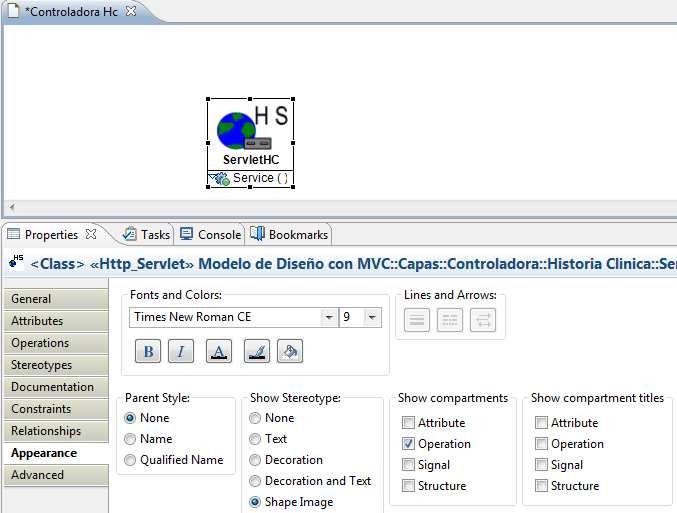
**2**

Poner como nombre a la clase ServletHC , crear la Operación Service y cambiar la apariencia a shape Image



**3**

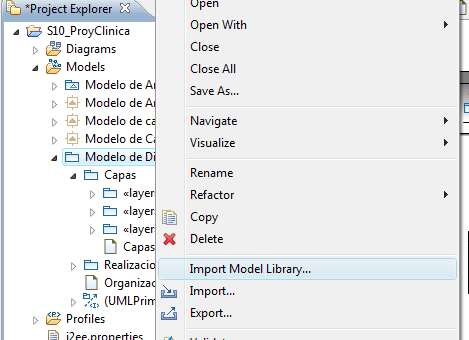
**4**



**5**

**6**

**Paso 2** Importar la librería de datos de Java RoseJavaDatatypes

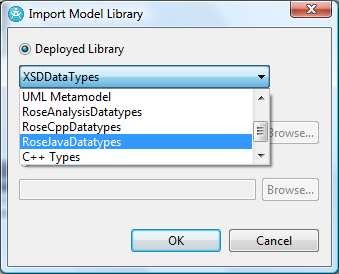


Click derecho sobre el Modelo de Diseño

**1**

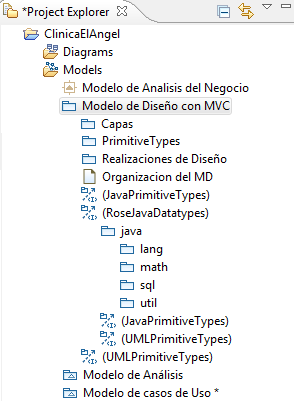
**2**

### Seleccionamos la librería RoseJavaDatatypes



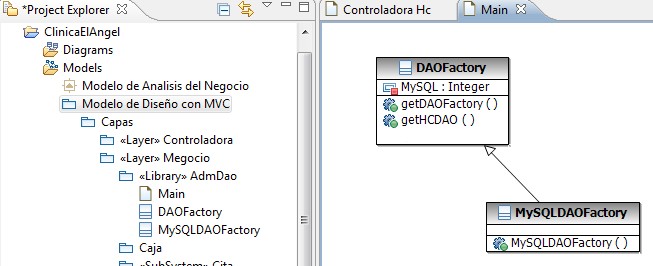
**3**

**4**

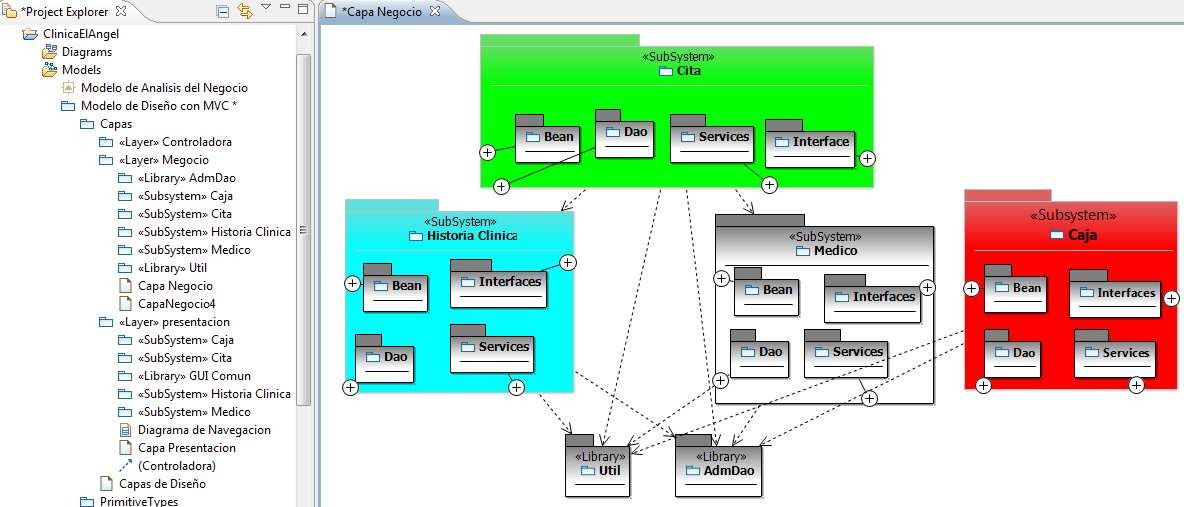


**5**

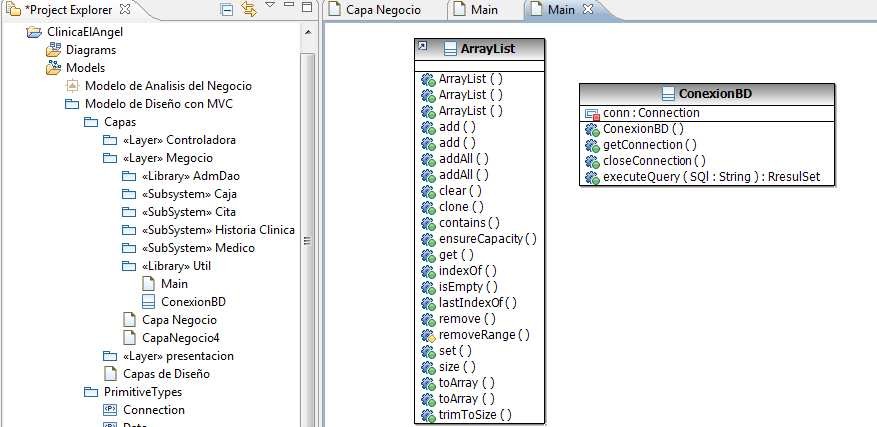
**PASO 3:** Crear las clases de la capa de Negocio, del paquete ADMDAO DAOFactory y MySQLDAOFactory en el Main del paquete AdmDAO:



Paso 4 Crearemos en la capa de Negocio para cada SubSystem los paquetes **Bean,** Dao, Interface y Services

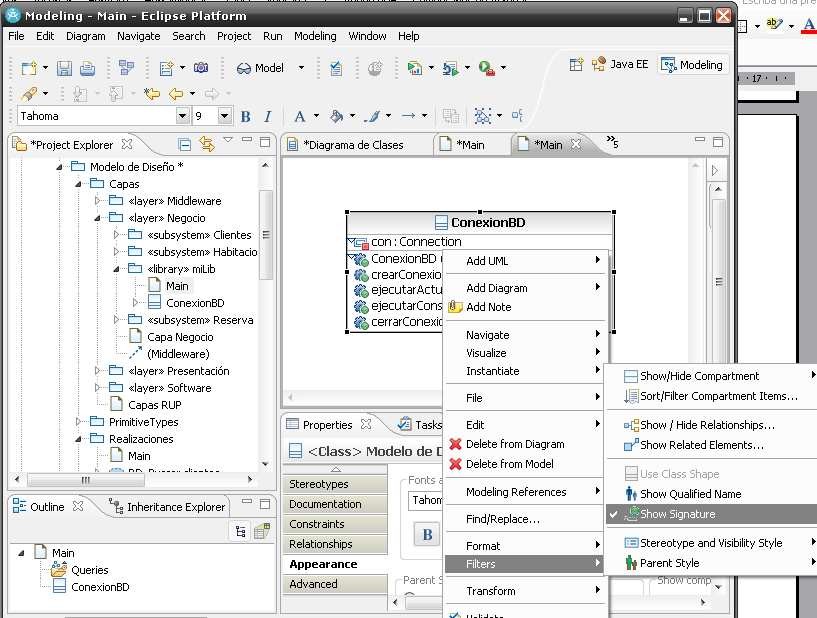


**PASO 5:** Cree la clase ConexionBD en el Main del paquete Util y arrastrar el arraylist



**NOTA:** Para mostrar la signatura completa de cada operación realice lo siguiente

1.

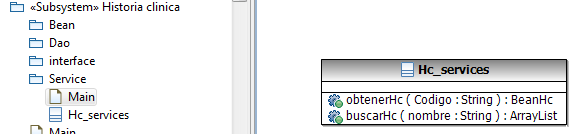
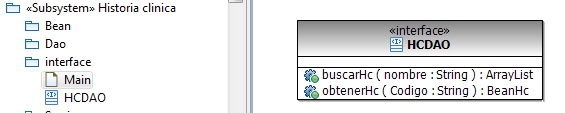
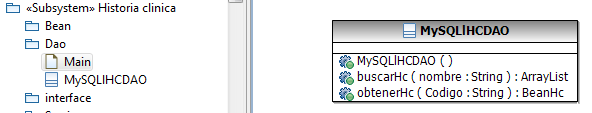
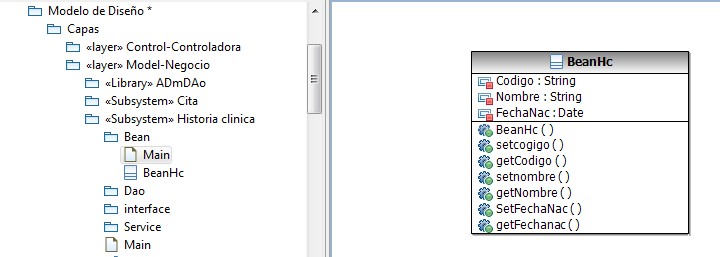


**1**

Clic derecho sobre la clase.

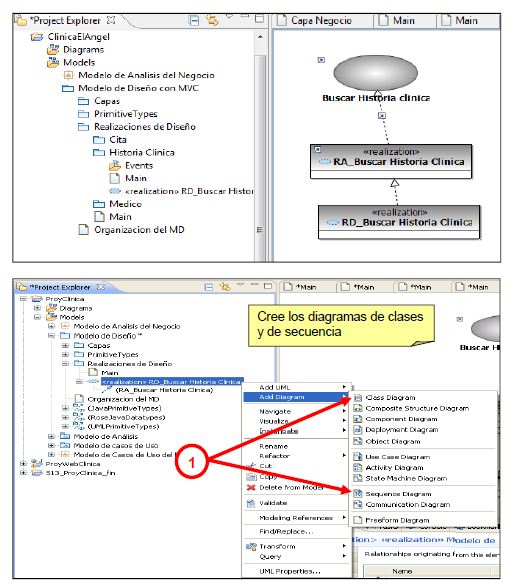
**2**

### PASO 6: Tenemos que crear el Bean, Dao, Interface y service de la clase Historia Clínica de la capa Model



Hasta este momento hemos creado los elementos de diseño que van a participar en el caso de uso “Buscar Historia Clínica”, ahora veremos las Realizaciones de diseño.

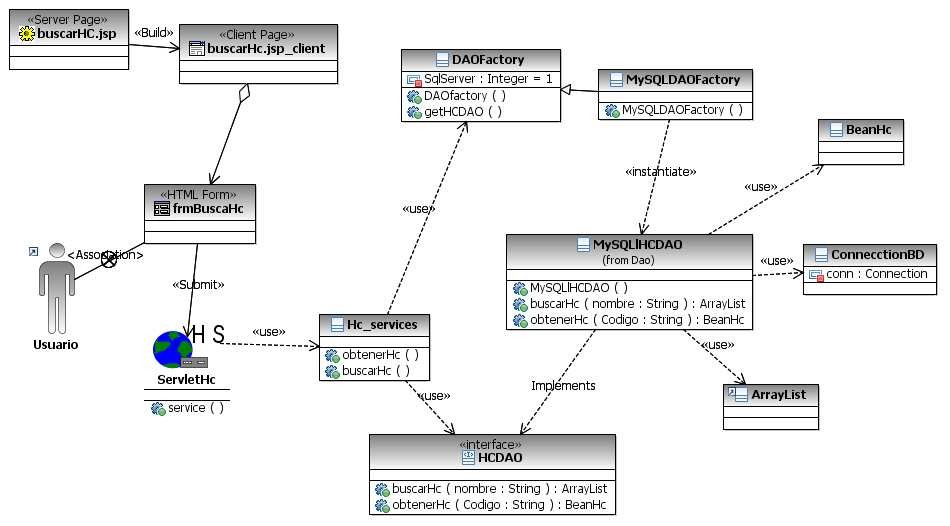
**PASO 7:** Ahora crearemos en el paquete de Realizaciones de Diseño, los diagramas de clases y diagrama de secuencias del flujo básico a partir de la realización de diseño buscar historia clínica del paquete Historia Clínica:



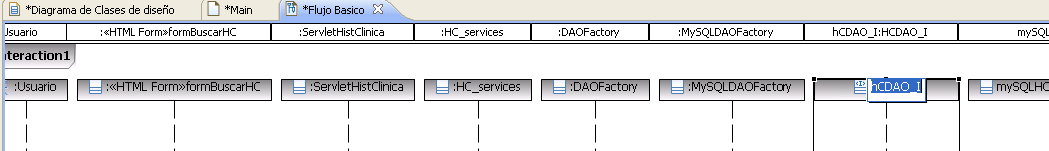
**PASO 8:** Arrastre el actor y las clases de las capas Presentación, negocio y Controladora.

* Es recomendable que las tres clases del JSP sean copiadas del Main de la capa Presentación a este Diagrama de clases.
* La clase ArrayList arrástrelo del Modelo de librería importado que se encuentra en la (RosejavaDataTypes) en / Java / Util.

Diagrama de clases de diseño



**PASO 9:** A continuación, en el diagrama de secuencia, arrastre las clases al flujo básico y, por cada elemento, dé clic sobre el texto para eliminar el nombre de instancia. Por cada método, se crea otro diagrama de secuencia.



Clic sobre el texto

del objeto.

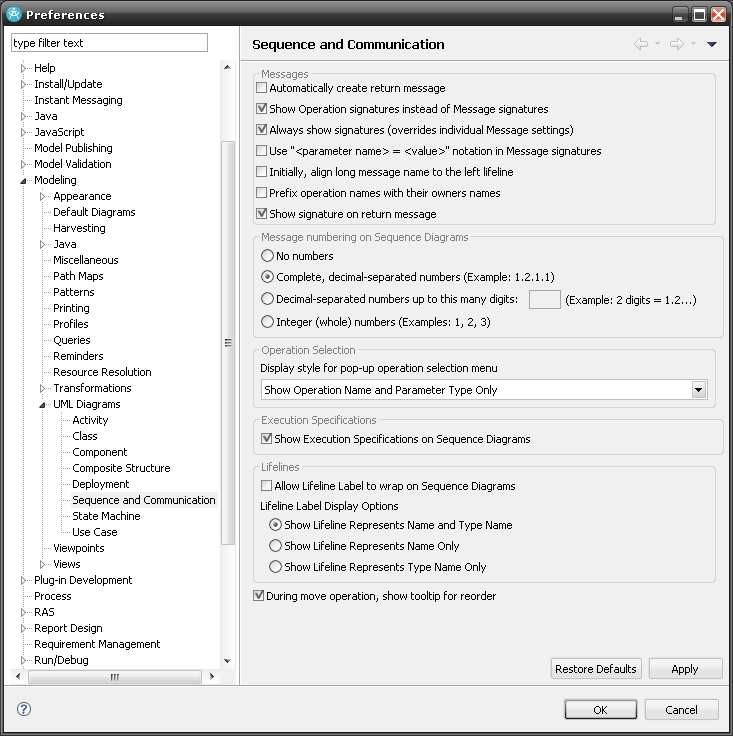
**1**

Borre el texto.

**2**

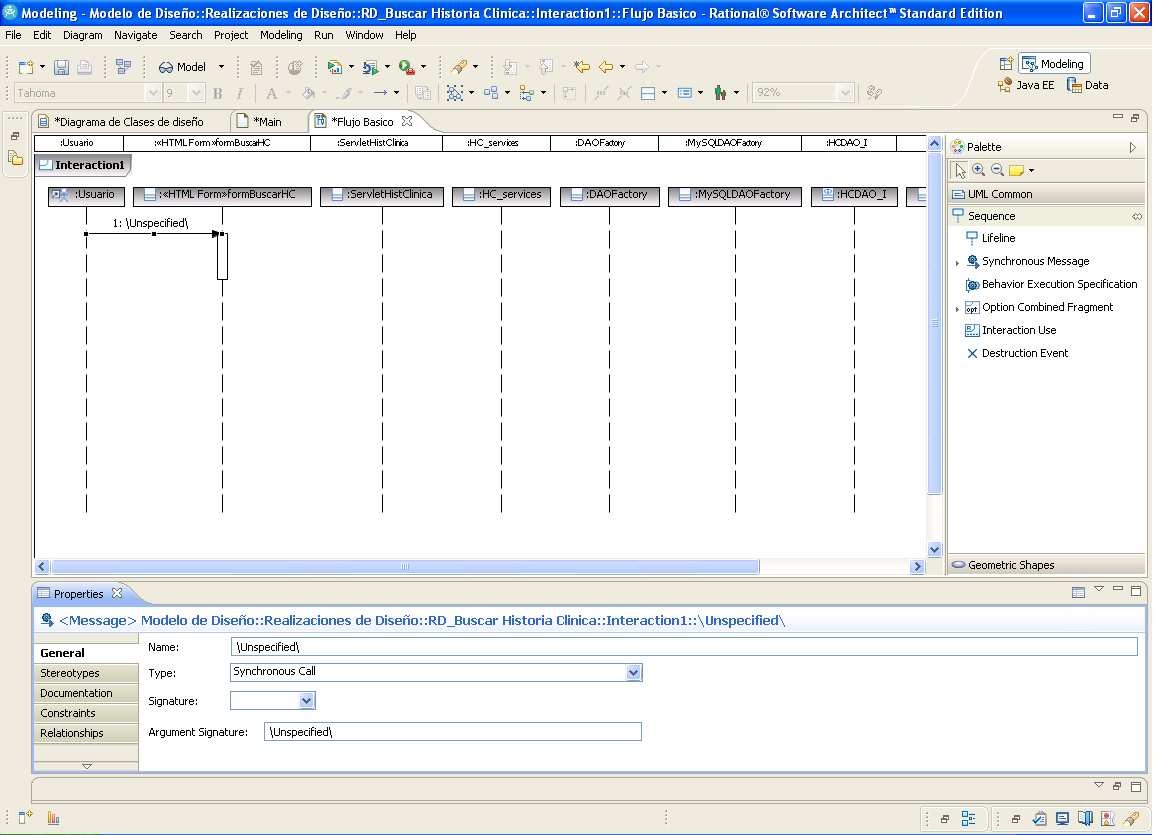
**PASO 10:** Configure el diagrama de secuencia para no mostrar mensajes de retorno:





Deje activo las opciones que se indican en la siguiente ventana:

**PASO 11** Agregue los mensajes síncronos. En caso no sea un método, edite el mensaje utilizando la caja de **Propiedades/General**:



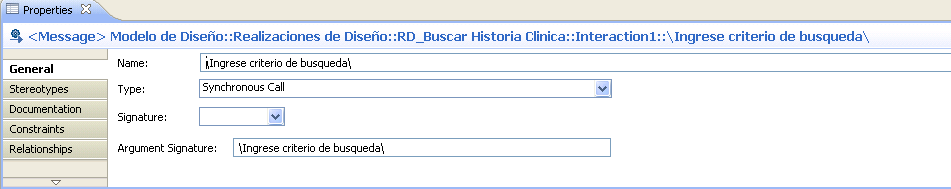
**2**

**1**

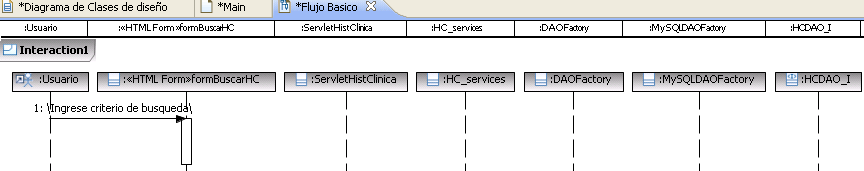
**3**

Edite el mensaje,

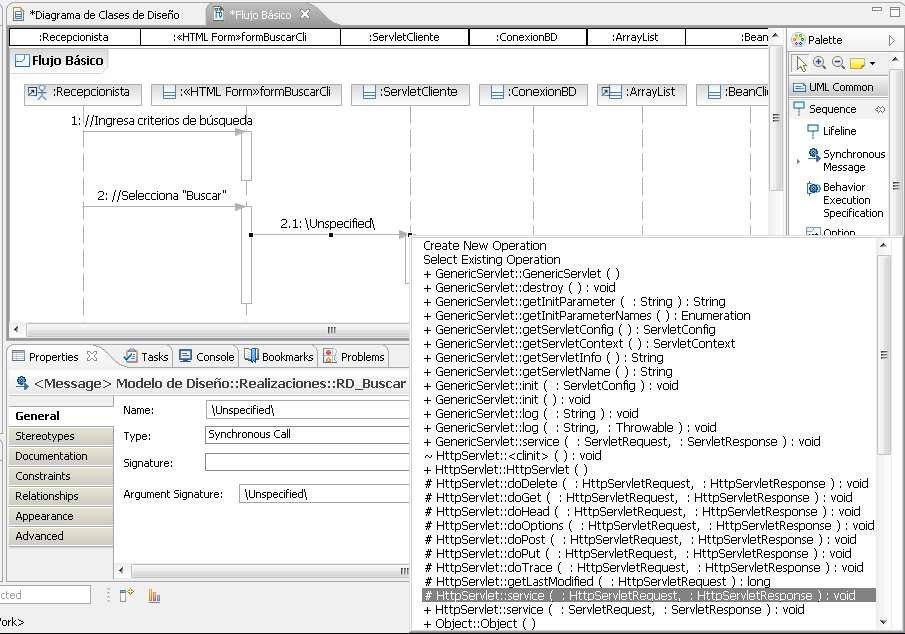
luego presione ENTER.



Se verá así:



**PASO 12:** Si el mensaje es un método del objeto destino, realice lo siguiente:

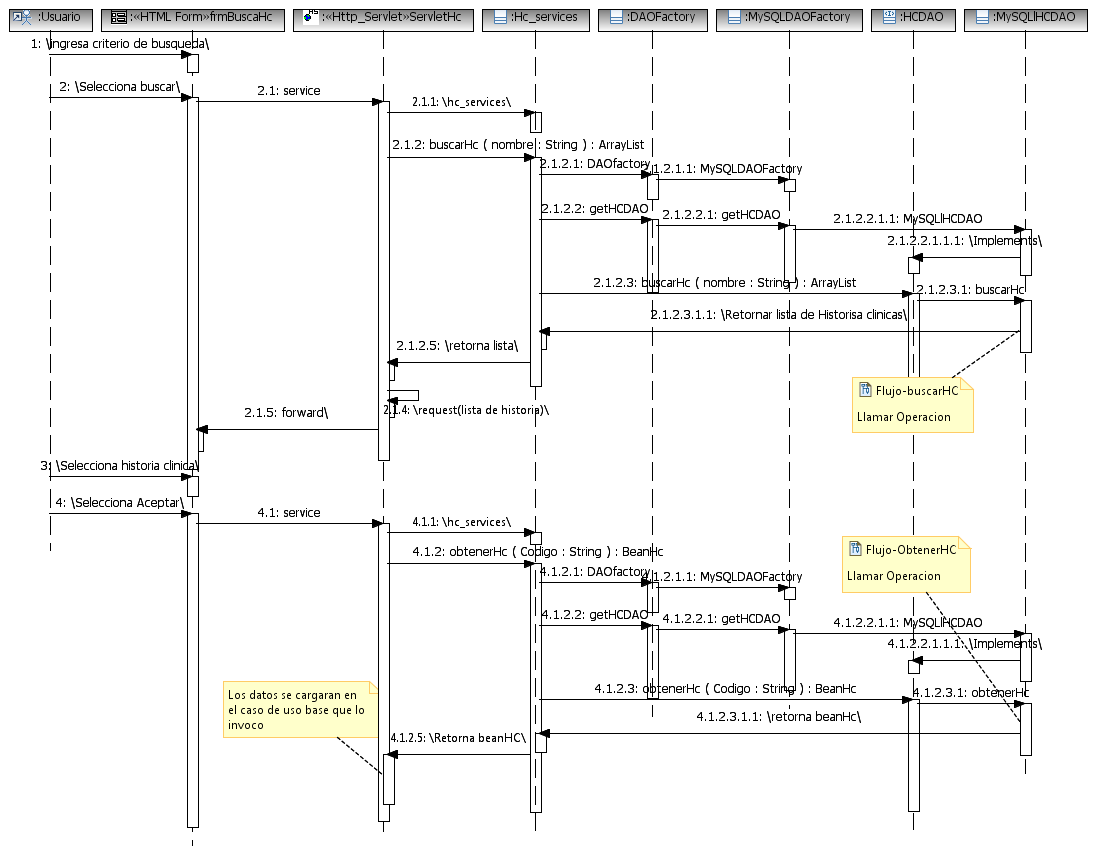


**1**

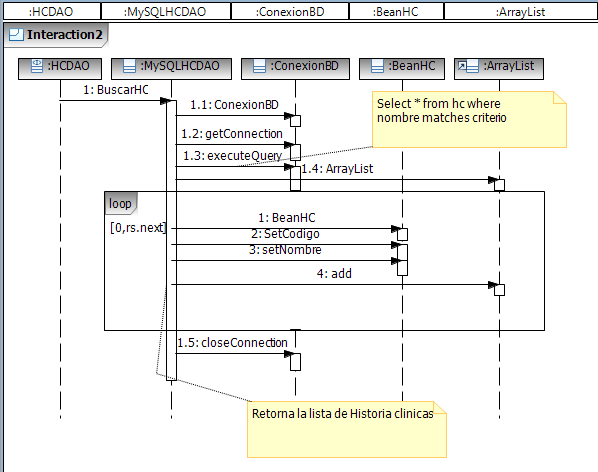
**2**

**3**

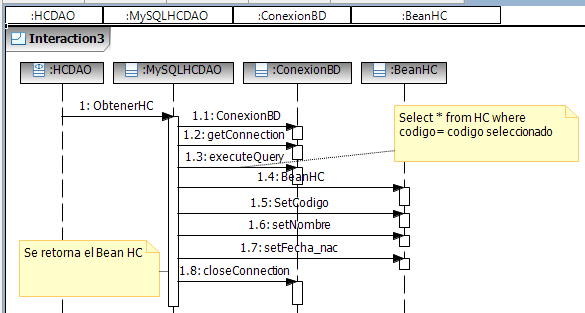
**PASO 13:** El diagrama de secuencia final para el flujo básico del CU Buscar Historia Clínica, es el que se muestra a continuación:



:**Paso 14** Flujo Buscar Buscar\_HC



:**Paso 14** Flujo Buscar Obtener\_HC



# CASO PRÁCTICO Nº 1

A partir de la Especificación de Caso de Uso, realice los siguientes artefactos:

1. Realización de Diseño del caso de uso
2. Diagrama de clases de diseño
3. Diagrama de secuencia del flujo básico
4. Diagrama de secuencia de las Operaciones

# Especificación de caso de uso: Generar Cita

### Descripción:

El caso de uso permite a la recepcionista de la clínica, registrar una cita médica para consultas externas por especialidad.

### Actor(es)

Recepcionista

### Flujo de Eventos

* 1. **Flujo Básico**
     1. El caso de uso comienza cuando la recepcionista selecciona la opción “Generar Citas” de la interfaz del menú principal.
     2. El sistema muestra la interfaz “GENERAR CITAS” con la fecha y hora de registro cargado y los siguientes campos:
        + Datos de la HC: número, nombre y apellidos del paciente.
        + Datos del médico: nombres, apellidos y especialidad.
        + Datos de la cita : fecha y hora de la cita y consultorio.
        + Además, presenta las opciones: Buscar HC, Buscar Horarios de Médico y Grabar Cita.
     3. La recepcionista selecciona “Buscar HC”.

#### El sistema incluye el caso de uso Buscar Historia Clínica.

* + 1. El sistema muestra los datos de la historia clínica del paciente.
    2. La recepcionista selecciona “Buscar Médico”.

#### El sistema incluye el caso de uso Buscar Horarios de Médico.

* + 1. El sistema muestra los datos del médico y de la cita.
    2. La recepcionista selecciona “Grabar Cita”.
    3. El sistema valida los datos.
    4. El sistema genera el número de cita y registra la cita con estado pendiente.
    5. El sistema muestra el mensaje “Cita generada”. El caso de uso termina.

### Flujos Alternativos

* + 1. **No existe HC**

Si en el paso 6 el sistema detecta que no existe la HC del paciente, muestra el MSG “No existe HC” y ofrece la posibilidad de registrar la HC del paciente.

### No hay médicos disponibles

Si en el paso 8 el sistema detecta que no hay médicos disponibles, muestra el MSG “No hay médicos disponibles” y el caso de uso finaliza.



**Grabar**

**3. Campos vacíos**

Si en el paso 10 el sistema detecta que alguno de los campos está vacío, muestra el MSG “Alguno de los campos está vacío” y el caso de uso continúa.

1. **Precondiciones**
   1. La recepcionista está identificada en el sistema.
   2. Lista disponible de historias clínica.
   3. Lista disponible de médicos.
2. **Poscondiciones**

2. En el sistema quedará registrada la cita en estado pendiente.

1. **Puntos de Extensión**

En el paso 6, el sistema extiende al caso de uso Mantener HC – subflujo “Registrar HC”.

1. **Requisitos Especiales**

Ninguno.

1. **Prototipos**

**Generar Citas**

**Fecha:** 08/03/2010

**Hora:** 15:32

**Datos de la HC del paciente Nº HC:**

**Paciente:**

**Datos del médico**

**Nombre y apellidos: Especialidad:**

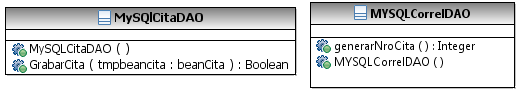
**Datos de la cita**

**Fecha:**

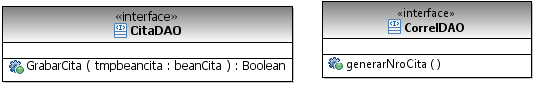
**Hora: Consultorio:**

**Paso 1:** Crear las clases de Negocio en el paquete Cita

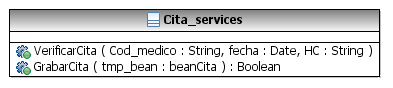
Crear las clases DAO MySQlCitaDAO y MYSQLCorrelDAO y BeanCita



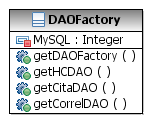
Crear las clases Interfaces CitaDAO y CorrelDAO



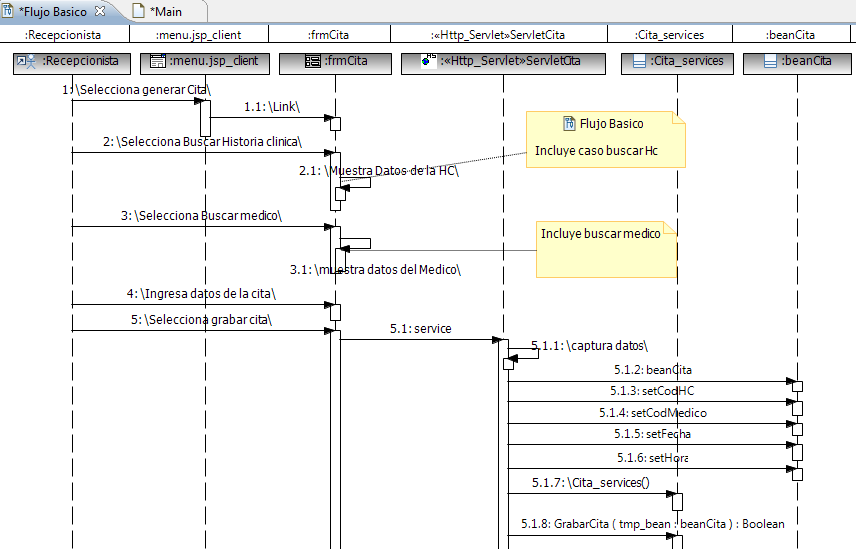
Crear la clase cita\_services

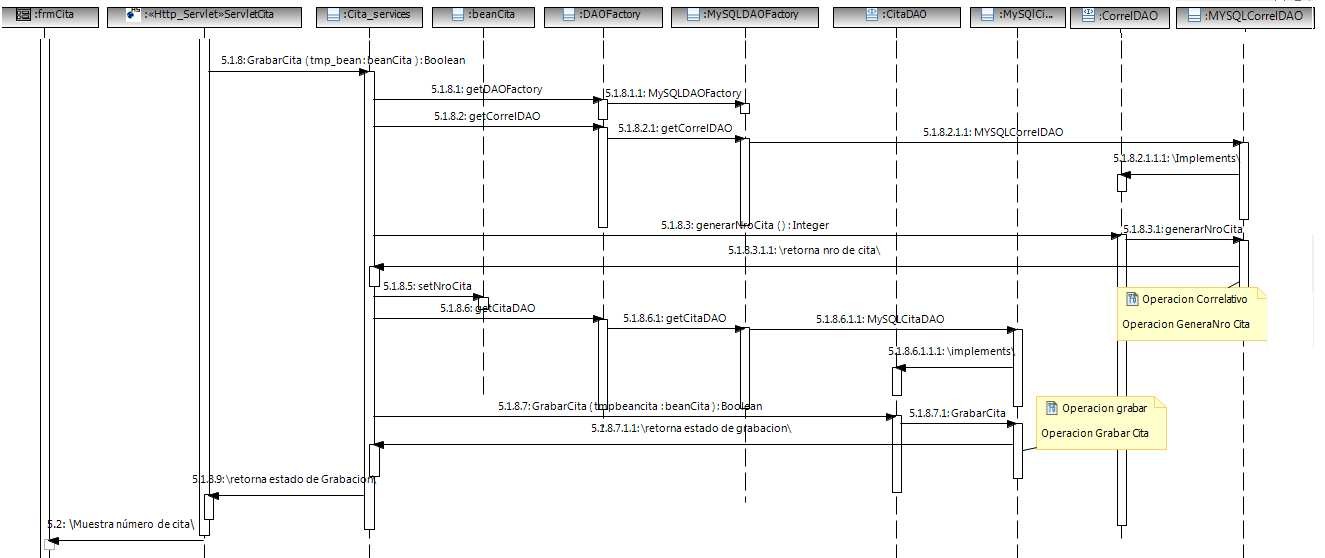


Agregar los métodos al DAOFactory

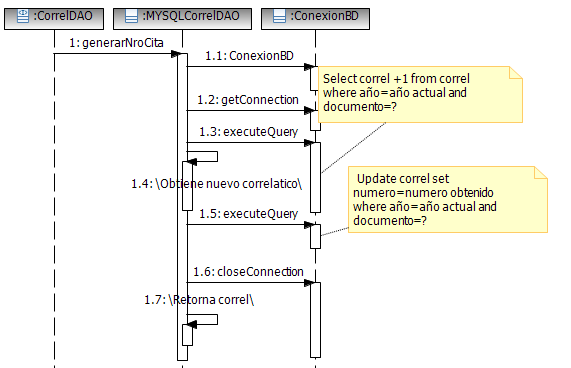


Flujo Básico

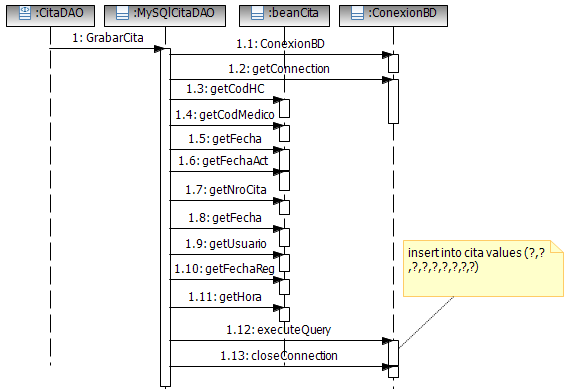




Operación generaNroCita



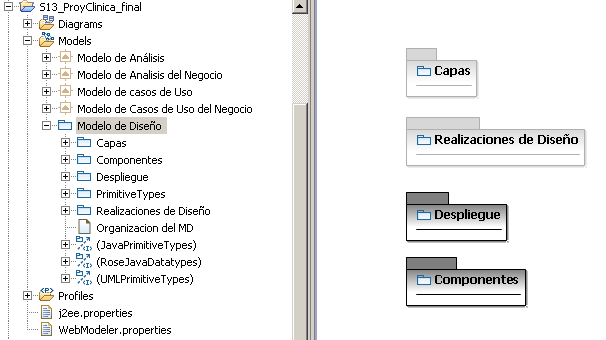
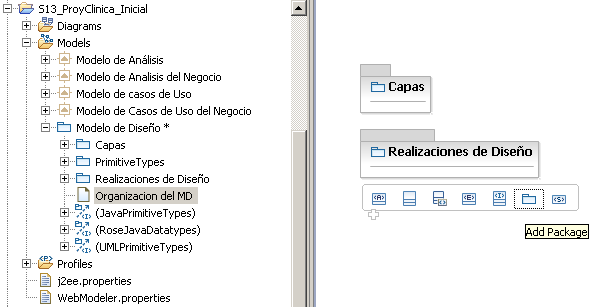
Operación grabar



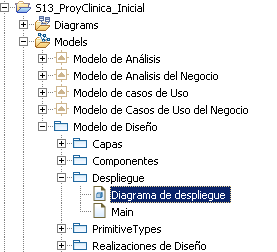
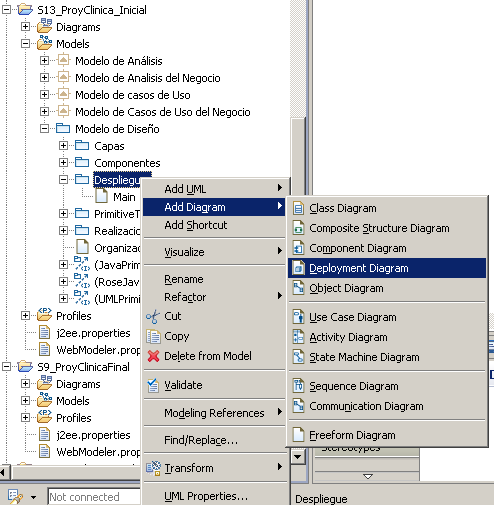
**5 Modelo de Diseño**

### Diagrama de componentes y despliegue

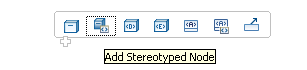
**PASO 1:** Crear 2 paquetes, uno para Despliegue y otro para Componentes en el modelo de diseño en la organización del MD.

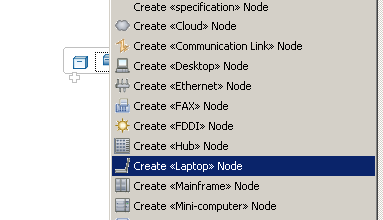


Paso 2: Agregar un diagrama de despliegue en el paquete Despliegue

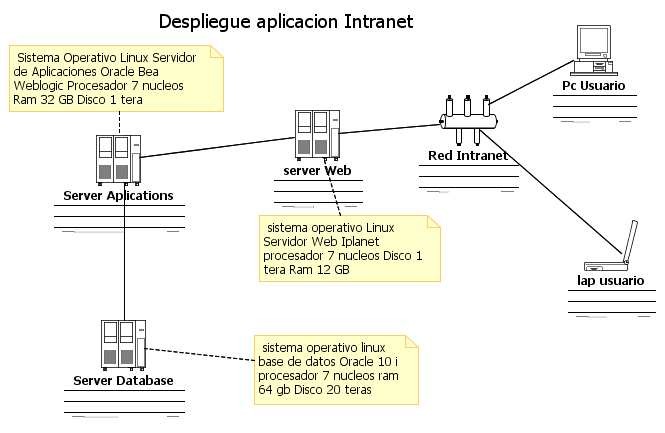


PASO 3: Agregar los nodos necesarios para armar la arquitectura de una intranet y colocamos en mensajes la especificación de cada nodo.

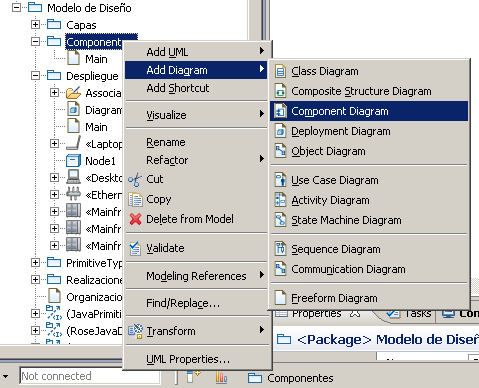




**PASO 4**: Relacionar los Nodos con el Communication PATH, quedando así:

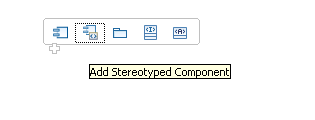


PASO 5: Agregar en el paquete de componentes un diagrama de componentes, el cual llamamos diagrama de componentes.

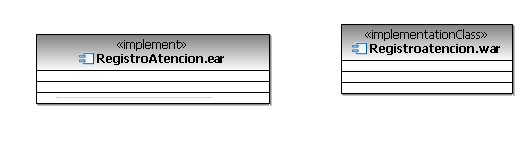


**PASO 6:** Agregamos los componentes necesarios. A continuación, detallamos cómo agregar componentes:

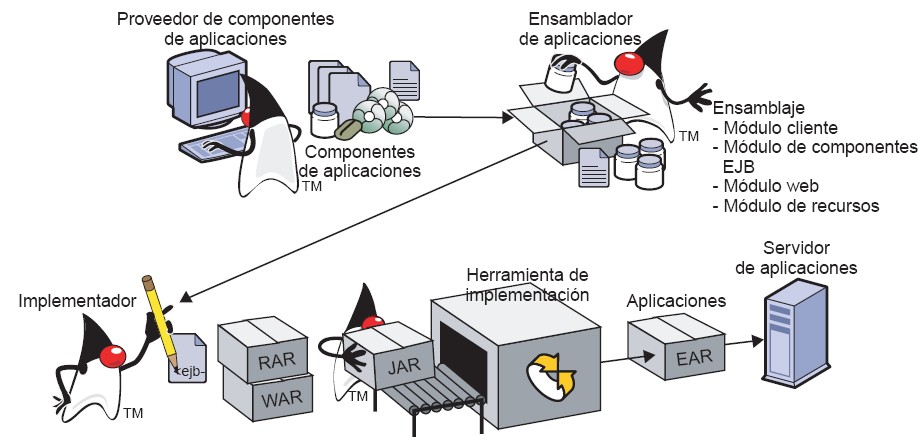
Seleccionamos el estereotipo de componente



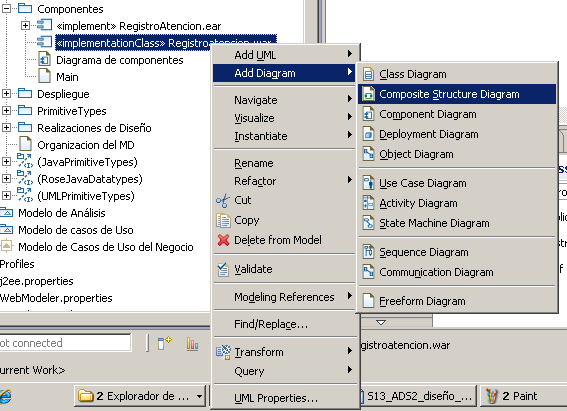
**Paso 7:** Agregamos 2 componentes 1 componente RegistroAtencion.war RegistroAtencion.ear.



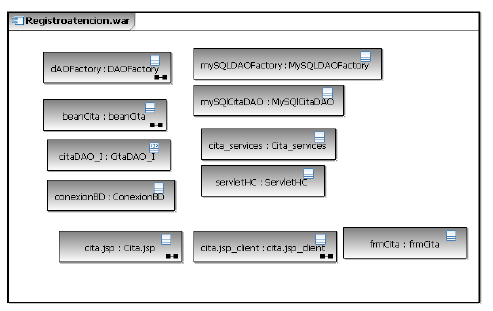
Basados en el concepto de instalación de JEE el War contiene todas las clases y dentro del ear colocamos solo al war



**PASO 8:** Dentro del componente RegistroAtencion.war creamos el diagrama de estructura compuesta y ahí colocamos las clases del proyecto.



**Paso 10**: Arrastramos todas las clases al diagrama de implementación creado.



Paso 11: Mostramos el RegistroAtencion.war con los elementos, seleccionando apariencia y el *check attribute*, y en el *ear* creamos lo mismo y agregamos el war.

