Casa Abierta al Tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD IZTAPALAPA

División de Ciencias Básicas e Ingeniería

Departamento de Ingeniería Eléctrica

Educación a Distancia

Reporte de Proyecto de Investigación I y II

Tesis que presenta el alumno(a)

Sandra Aide Valadez López
94319656

Para obtener el grado de :

Licenciatura en Computación

Asesor

M. en I. David Uzziel López Illescas

Septiembre 2000.

Gracias a Dios,
por haberme dado la oportunidad de
conocer a una persona muy especial en mi vida.

A Enrique Cabildo Velázquez
por su tiempo, su paciencia, su ayuda y apoyo,
por sus grandes enseñanzas, pero sobretodo
por su gran inteligencia.

TABLA DE CONTENIDO.

1. EDUCACION A DISTANCIA.

Este proyecto surgió con el fin de poder dar cursos o clases de Posgrado vía Internet en el área de Ciencias Sociales y Humanidades de la Universidad Autónoma Metropolitana Iztapalapa. Esta idea fue propuesta por la division de Ciencias Sociales y Humanidades en conjunto con el Laboratorio de Ingenieria de Software (LIS) area de proyectos de software.

Este proyecto nace para facilitar la difusión de clases o cursos de Posgrado, dar información en la red de dichas clases o cursos, además para tener la opción de poder tomar una clase fuera de las aulas, así como poder tomar el curso desde cualquier lugar en el que se cuente con Internet.

Es necesario destacar que dicho proyecto se creo como ya mencionamos para impartir clases del área de Ciencias Sociales pero se puede ampliar a otras áreas.

Sobre el documento:

El siguiente documento muestra; algunas técnicas utilizadas para la elaboración del proyecto, como lo es el Análisis y Diseño Orientado a Objetos, UML, además de la programación que se realizo en Java, así como referencias a libros o páginas utilizadas.

También se hace mención en aquellas cosas que no se toman en cuenta en los manuales y se llegan a descubrir cuando se programa. Así como problemas con los que se encuentra el programador en la práctica.

2. OBJETIVOS DEL SISTEMA.

Objetivos Generales:

- Elaborar una aplicación Cliente-Servidor vía Internet donde los maestros puedan dar clases desde su máquina, y los alumnos se puedan conectar a esta.
- Las clases se darán a través del envío y recepción de texto.
  así como el envió y la recepción de imágenes.
- Recepción de preguntas por parte de los alumnos al profesor.
- El maestro podrá enviar texto.
- El alumno podrá enviar texto.

Objetivos Específicos:

- La aplicación debe ser amigable
- La programación será en Java, haciendo uso de applets y servlets.
- Se crearan hojas en HTML, para presentarlas en la RED.
- Cada alumno tendrá una clave de acceso para dicha clase.
- Los alumnos podrán enviar preguntas a los maestros.
- No existirá la comunicación entre los alumnos, únicamente por medio del profesor.
- El maestro podrá abrir archivos que podrá enviar a los alumnos tanto a la pantalla de exposición como a páginas HTML ya previamente establecidas para éstos fines.
- El maestro puede enviar preguntas o comentarios a los alumnos.
- El maestro podrá ver que alumnos están conectados, así como las preguntas que le envíen.

3. MARCO TEÓRICO

A continuación se citan las técnicas utilizadas para el desarrollo del proyecto. Así como sus puntos finos en la construcción, de la cual se hace un énfasis especial en el análisis y diseño orientado a objetos porque ha sido la base de todo el desarrollo del proyecto.

3.1 Técnicas utilizadas en el desarrollo del proyecto.

Las principales técnicas utilizadas para el desarrollo del proyecto, son en la Ingeniería de Software el análisis de riesgos, el análisis y el diseño orientado a objetos, el desarrollo por prototipos, estrategias para tratar el riesgo, la codificación y las pruebas.

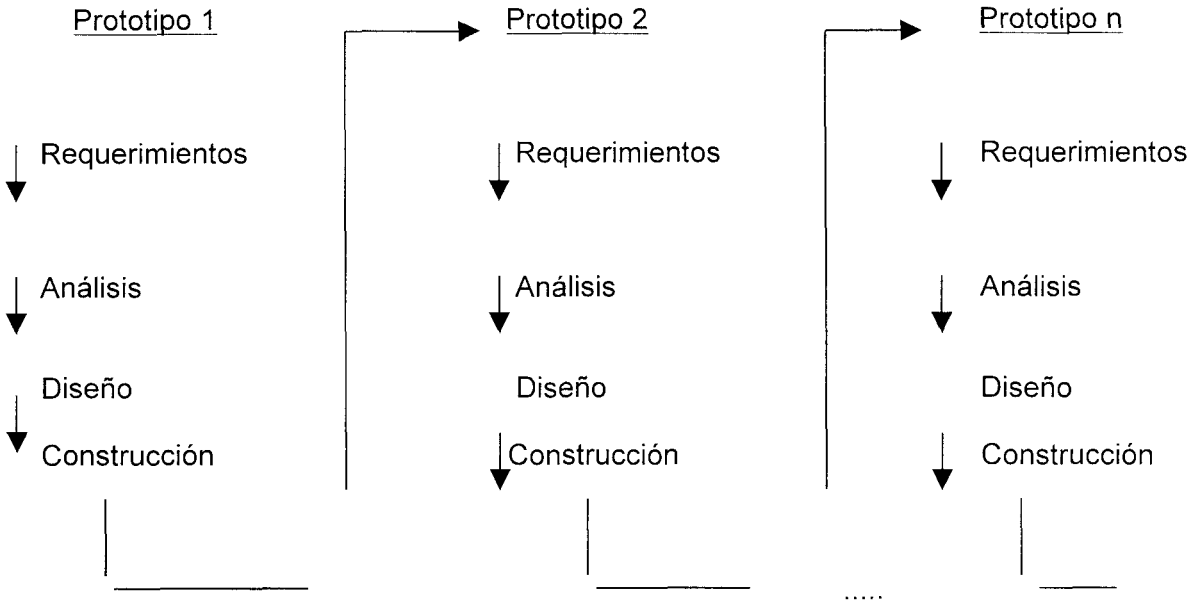
3.1.1 - Análisis y diseño orientado a objetos.

El proyecto se desarrolló sobre la base de una metodología de Análisis y Diseño Orientada a Objetos, que tiene como meta el desarrollo de prototipos incrementales, es decir, partiendo de un prototipo 0 con solamente unas cuantas funciones básicas, se van generando prototipos subsecuentes con mayor funcionalidad, cada uno de los cuales es alimentado por el anterior. El desarrollo incremental por prototipos minimiza el riesgo de entregar un producto con errores en todos sentidos, pues cada uno de ellos paulatinamente incrementa la utilidad del sistema y permite la revisión por parte de los usuarios desde etapas tempranas. Así, el cliente retroalimenta el desarrollo participando activamente como un aliado y no como un enemigo a vencer.

El análisis y diseño orientado a objetos se divide en varias etapas, cada una con entregables bien específicos que alimentan a la etapa siguiente. Estas etapas son:

Etapa	Entregable
1. - Levantamiento de Requerimientos.	Modelo de requerimientos. Modelo de Casos(USE CASE). Modelo de Interfaz: Pantallas principales. Diagramas de transición de estados. Modelo de dominio del problema.
2. - Análisis.	Modelo de Análisis.
3. - Diseño.	Modelo de Diseño.
4. - Construcción.	Modelo de Construcción.

Cada uno de los prototipos se desarrolla siguiendo estas etapas en estricto orden, puesto que cada una alimenta a la siguiente. Por lo tanto, cada uno de los prototipos es un micro-desarrollo, como se muestra en la fig1.



3.1.1.1 Análisis de Requerimientos.

Esta etapa se encarga de levantar los requerimientos y especificaciones de los usuarios, de modo que queden lo suficientemente claros para evitar errores en las etapas posteriores. Esta parte es crucial porque los requerimientos obtenidos alimentarán todo el desarrollo posterior, y entonces se debe garantizar que lo que el usuario desea sea lo que el desarrollador entiende, con lo que se tiene una base confiable. Aparte de estas importantes implicaciones, el levantamiento de requerimientos sirve para que el equipo de desarrollo vaya adquiriendo un conocimiento aceptable del entorno sobre el que va a trabajar, pues son especialistas en desarrollar sistemas, no en la materia sobre la cual va a desarrollar. El aprendizaje del entorno permite la elaboración de un producto más eficiente y libre de errores, pues así desarrolla una solución de un problema que conoce (o que va conociendo). Todo este conocimiento lo absorbe de usuarios clave, que conocen lo más perfectamente posible el problema. El entregable aquí es el modelo de requerimientos, que se constituye a su vez de tres modelos:

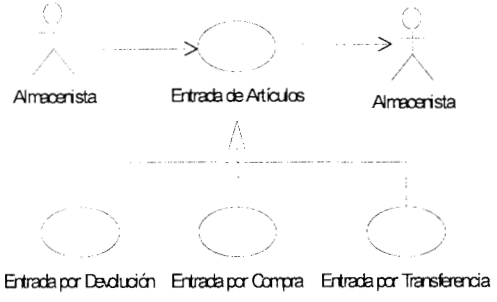
- Modelo de Casos
- Modelo de Interfaz
- Modelo del Dominio del Problema

El ***Modelo de Casos*** extrae el conocimiento funcional fundamental del problema de una forma estructurada y progresiva, siendo la base para establecer la estructura del sistema. Este modelo orienta todos los demás procesos del método. Para plasmar exactamente lo que hace un sistema, se necesita definir las transacciones principales que lo componen, como se comunican y el orden que siguen para describir su funcionalidad.

3.1.1.2 Ejemplo de Use Case.

Un USE CASE es precisamente una descripción de una transacción, donde no sólo exhibe lo que hace, sino también incluye una serie estructurada de pasos para realizarla y los entes que interactuan con él o Actores. Cada USE CASE se trata por separado como un **objeto.**

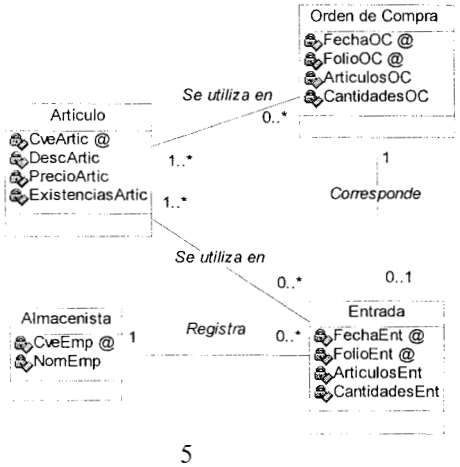
El sistema completo puede constar de una infinidad de transacciones, pero se ordenaran con una jerarquía de herencia, en la que cada nivel hará uso o derivara a su vez otras transacciones o USE CASES hasta describir la totalidad, siempre tratando de guardar la proporción de 5-9 o ± 7 objetos en cada nivel para asegurar la percepción(una cantidad mayor será difícil de entender, una cantidad menor podría ser incluida en otro nivel).



Ejemplo de USE CASE.

Sobre el *Modelo de Interfaz* establece el vínculo visual entre el desarrollador y el usuario para concretar aspectos de la interacción que el sistema pudiese tener con su entorno externo, permitiendo la retroalimentación temprana de aspectos fundamentales para el conocimiento de la aplicación. Consta de las pantallas principales que se utilizarán al desarrollar una transacción y su correspondiente Diagrama de transición de estados.

Sobre el *Modelo del Dominio* del Problema se establecerán los principales objetos que constituirán al sistema y las relaciones que tienen entre sí. Los objetos persistentes almacenarán información, es decir, la base de datos real, que después se tendrá que pasar irremediablemente a un modelo relacional y no orientado a objetos; por lo que las relaciones obtenidas aquí serán muy útiles.



Ejemplo de Modelo de dominio del problema

El proceso completo de levantamiento de requerimientos se desarrolla de la siguiente manera:

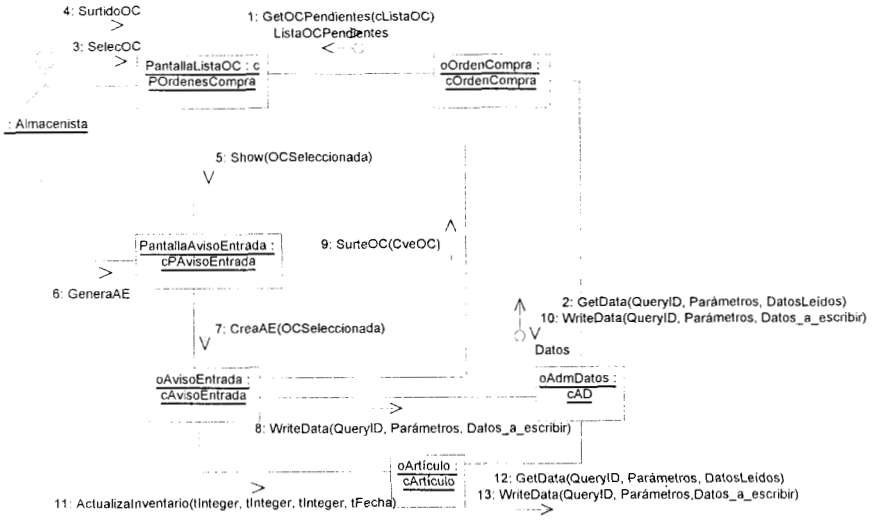
Inicialmente se entrevista a los usuarios principales o con poder de decisión de la empresa y se les pide que definan las transacciones principales que definen al sistema, se tratarán de acotar a solamente aquellas de las cuales no se pueda prescindir y sobre las que se enfocará el mayor esfuerzo. Después se convendrá una cita con los usuarios más expertos en las transacciones principales para que describa los pasos a realizar para llevarlas a cabo. Se desarrolla un modelo inicial con él o los USE CASE que se obtuvieron, pantallas iniciales y demás componentes. En citas posteriores se propondrán estas pantallas y descripciones al usuario y se refinarán con las observaciones que haga. Esta etapa termina cuando los cambios que se hagan al modelo sean mínimos.

Es importante recalcar que el procedimiento más pesado de levantamiento de requerimientos como se indica anteriormente, y el análisis y diseño se hace al principio del proyecto. En los prototipos subsecuentes se realiza también estas etapas para añadir nuevas transacciones y modificar las existentes, pero el trabajo ya no es tan completo.

Sobre el Análisis aquí se dan las relaciones que existen entre los diversos objetos encontrados y se diseñan nuevos que puedan ayudar a la tarea específica de cada uno. Al final se tendrá una descripción detallada del comportamiento y comunicación que existe entre los diversos objetos que realizan una transacción. Para entender mejor estas relaciones se construyen escenarios, que son instanciaciones de una transacción, es decir, descripción de los pasos necesarios y los valores reales que se obtienen o se transmiten. Este escenario es referente a un comportamiento perfecto, y los errores y excepciones se toman en cuenta hasta que un escenario es perfectamente interpretado. Aquí es importante recalcar la utilidad del modelo de tres capas. En dicho modelo todos los objetos que interactuan en una transacción se dividen en tres grupos de acuerdo a su funcionalidad: los objetos de interfaz que solamente se encargan de capturar o exhibir información, los objetos de negocio que manejan la lógica para manipular y procesar datos, ya sea para entregarlos a un objeto de interfaz o para obtenerlos de un objeto de datos, cuya única función es servir como un mensajero eficiente hacia o desde la base de datos permanente. Al separar los objetos en estos tipos su ordenamiento es adecuado y el mantenimiento es más sencillo.

El producto principal de este proceso es el diagrama de colaboración, que describe todos los objetos en juego para cada transacción, las llamadas que reciben y hacen hacia otros objetos con sus parámetros respectivos(con sus tipos de datos recibidos y a retornar).

El diagrama de colaboración debe ser lo suficientemente detallado para entender estas relaciones.



Ejemplo de Diagrama de Colaboración.

3.1.1.3 Diseño.

Prácticamente es una extensión al análisis con la adición de que contempla aspectos de implantación, es decir, toma en cuenta las características especiales de la plataforma en que va a utilizarse para decidir que tipo de datos se utilizara, el tipo de estructuras de almacenamiento mas adecuadas y demás detalles característicos. El entregable es el diagrama de secuencia, que es una interpretación gráfica exacta y en orden estricto de todas las llamadas que hacen y reciben los objetos que realizan una transacción. Aquí todos los métodos de los distintos objetos tienen ya bien especificados sus tipos de datos en base a una plataforma dada. Estos diagramas son la especificación completa a seguir para desarrollar el producto y tiene que ser lo mas clara y precisa posible para que el programador la siga al pie de la letra.

3.1.1.4 Construcción.

Es ya la elaboración del producto. Culmina todos los esfuerzos con un prototipo funcional que sigue las especificaciones de todas las etapas anteriores, listo para probarse y modificarse alimentando un prototipo subsecuente. Para un buen entregable habrá que colocar comentarios amplios en el código donde hagan falta para facilitar el mantenimiento y tener un producto legible.

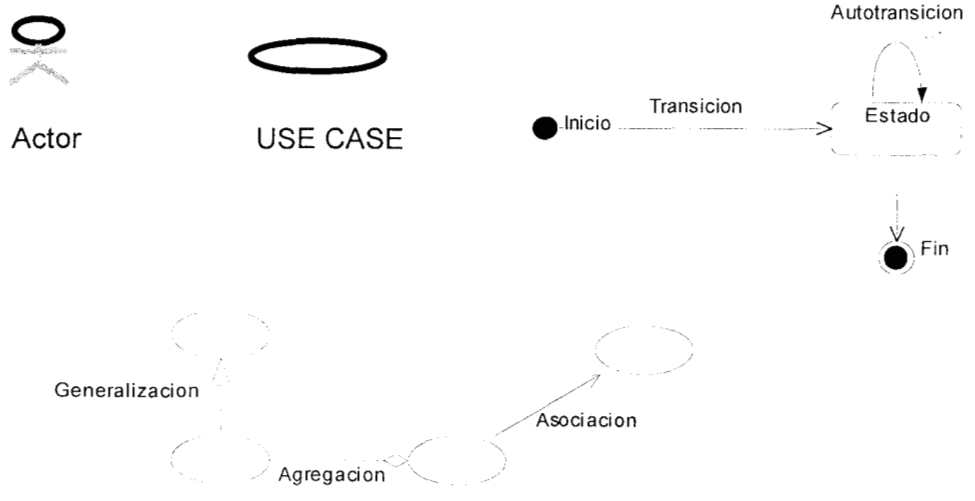
Cabe hacer notar que el proceso de pruebas se encuentra inmerso en todas las etapas anteriores y por eso no se le considera una más. Al realizar el levantamiento de

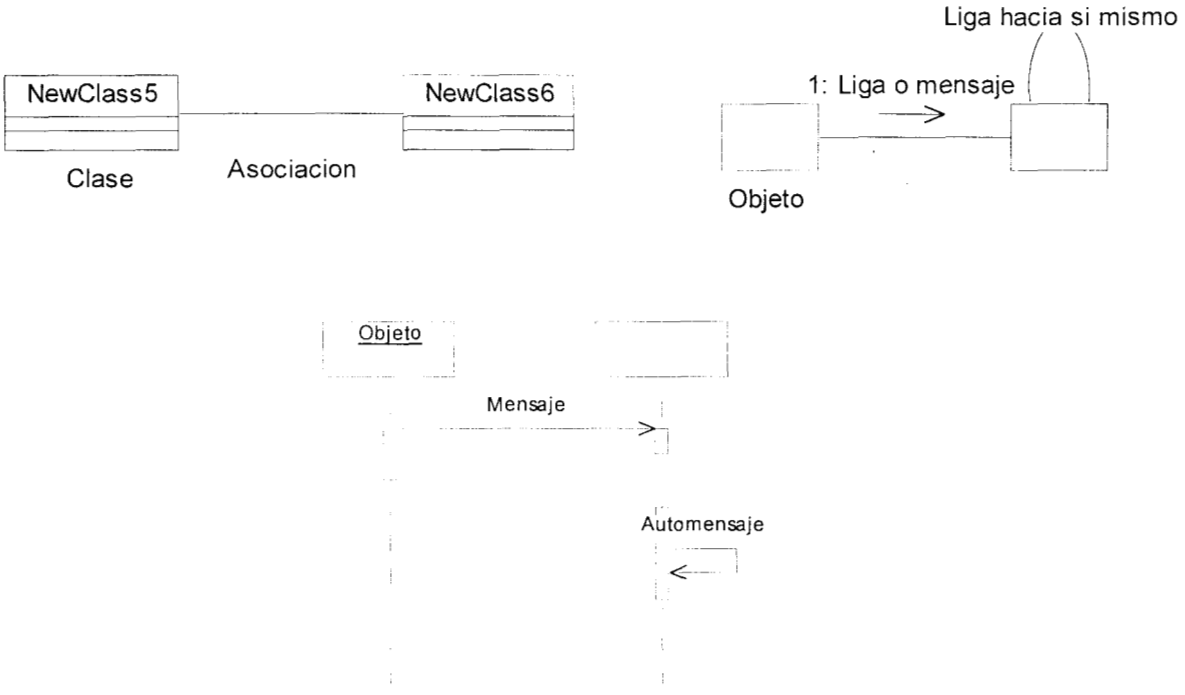
requerimientos en un principio los cambios son extensos por las observaciones y modificaciones propuestas por el usuario, tendiendo a estabilizarse, con lo cual el procedimiento de prueba se encuentra inmerso en el trabajo. Cuando los cambios son mínimos entonces el usuario ya ha dado de forma implícita una prueba y una aceptación, dando luz verde para el arranque de las demás etapas, alimentadas ya con requerimientos sólidos. Al desarrollar las demás etapas de análisis y diseño y explotar una transacción dada, al sobresalir algunas rectificaciones para integrar, al realizar pruebas de escritorio y resolver dudas de los programadores acerca de especificaciones, diseños y aspectos relacionados, se están realizando pruebas. Al entregar un prototipo para revisión la entrega misma es una prueba, con la ventaja de que se hace en etapas muy tempranas del desarrollo, permitiendo verificar y corregir a tiempo. En suma, durante todo el desarrollo se hacen pruebas, tomando como un hecho que una etapa alimenta a la siguiente con entregables sin errores en la medida de lo posible, y que además con esta metodología es sencillo corregir un fallo, pues nunca serán errores trascendentales.

Todas estas etapas son repetidas en el desarrollo de cada prototipo, que a su vez alimentara a uno posterior hasta que el nivel de errores y cambios sea bajo y se haya logrado atacar con eficacia las transacciones más importantes.

3.1.2.- UML

Para seguir una metodología orientada a objetos es indispensable un lenguaje standard bien establecido que permita modelar las ideas de una manera correcta y que permita su traslado a código sin errores. Para plasmar toda la información en cuanto a levantamiento de requerimientos, análisis y diseño se utilizo el lenguaje unificado de modelado o UML, que a ultimas fechas se ha convertido en un estándar en el área de análisis y diseño orientado a objetos. Los sencillos ejemplos mostrados en la sección anterior de diagramas de estados, colaboración y secuencia han sido elaborados con base en este lenguaje. Al ser un lenguaje estándar nos da un punto de partida ideal para que cualquier gente que tenga nociones de el entienda nuestras ideas y su contexto, transformándose en un medio de comunicación efectivo, sin tener que extenderse demasiado en explicaciones, ejemplos, etc. Algunos elementos más importantes del lenguaje son los sigs. :





3.1.3.- Rational Rose.

Para poder diseñar en base a un lenguaje se necesita una herramienta que facilite el trabajo. La que mejor se adapto a las necesidades y disposición que se tenia para el presente trabajo fue Rational Rose, una herramienta CASE que proporciona una interfaz gráfica para el modelado. Con ella se tiene una buena administración de los proyectos porque permite el desarrollo de prototipos incrementales y su uso es bastante sencillo. En el caso del proyecto el trabajo se dividió en partes que pudieran desarrollarse paralelamente, se analizo el resultado de cada una y al final se unieron en un proyecto principal. Cada cambio que se le hacia al diseño era efectuado en el proyecto de rose principal de manera que todo mundo los apreciara y pudiera modificarlo avisando desde luego de dichos cambios.

3.1.4.- Técnicas de programación en Java.

Como el proyecto esta basado en el uso de Internet el lenguaje ideal es Java. Este lenguaje es multiplataforma y el acceso a través de Internet tiene como único requisito el uso de un browser compatible con Java. Las ventajas son evidentes y se citan a continuación:

1. - Java es multiplataforma. Puede un mismo programa correr en diferentes maquinas sin compilar y modificar nuevamente.

2. - Al utilizarse paginas HTML dinámicas o Applets el código reside realmente en la maquina servidor(gateway) y es transportado e interpretado en la maquina cliente, así que para realizar modificaciones y mantenimiento se trabajará únicamente sobre el código que esta disponible en el servidor y no en todas las maquinas cliente.

3. - La programación del lado servidor en Java es sencilla y eficiente, ya que a diferencia de guiones CGI los servlets o clases especializadas en proporcionar servicios HTTP permanecen residentes en memoria, pueden mantener conexiones costosas con bases de datos y su programación no es tan laboriosa. Por otro lado, el uso de JDBC o conexión abierta con bases de datos, permite que su uso sea flexible a este respecto y tenga fácil mantenimiento. JDBC (Java DataBase Connectivity) es un API de Java que permite al programador ejecutar instrucciones en lenguaje estándar de acceso a Bases de Datos, SQL (Structured Query Language, lenguaje estructurado de consultas), que es un lenguaje de muy alto nivel que permite crear, examinar, manipular y gestionar Bases de Datos relacionales. Para que una aplicación pueda hacer operaciones en una Base de Datos, ha de tener una conexión con ella, que se establece a través de un driver, que convierte el lenguaje de alto nivel a sentencias de Base de Datos. Es decir, las tres acciones principales que realizará JDBC son las de establecer la conexión a una base de datos, ya sea remota o no; enviar sentencias SQL a esa base de datos y, en tercer lugar, procesar los resultados obtenidos de la base de datos.

4. - Como es un lenguaje 100% orientado a objetos, es el lenguaje más idóneo para un análisis y diseño correcto orientado a objetos.

5. - Como las clases predefinidas que posee tienen una alta orientación al uso de Internet, mucho del trabajo de bajo nivel y más complicado ya se encuentra resuelto y probado, con lo que se ahorra mucho tiempo en programación.

6.- El parecido con el lenguaje C es sorprendente, con lo que se facilita mucho el aprendizaje.

Java es el primer lenguaje que tiene la virtud de ser compilado e interpretado de forma simultánea. Cuando un programador realiza una aplicación o un applet en Java y lo compila, en realidad, el compilador no trabaja como un compilador de un lenguaje al uso. El compilador Java únicamente genera el denominado ByteCode. Este código es un código intermedio entre el lenguaje máquina del procesador y Java. Evidentemente este código no es ejecutable por sí mismo en ninguna plataforma hardware, pues no se corresponde con el lenguaje de ninguno de los procesadores que actualmente se conocen (habrá que esperar a ver qué ocurre con los procesadores Java). Por lo tanto, para ejecutar una aplicación Java es necesario disponer de un mecanismo que permita ejecutar el ByteCode. Este mecanismo es la denominada Máquina Virtual Java. En cada plataforma (Unix, Linux, Windows 95/NT,Macintosh, etc.) existe una máquina virtual específica. Así que cuando el ByteCode llega a la máquina virtual, ésta lo interpreta pasándolo a código máquina del procesador donde se esté trabajando, y ejecutando las instrucciones en lenguaje máquina que se deriven de la aplicación Java. De este modo, cuando el mismo ByteCode llega a diferentes plataformas, éste se ejecutará de forma correcta, pues en cada una de esas plataformas existirá la máquina virtual adecuada. Con este mecanismo se consigue la famosa multiplataforma de Java, que con sólo codificar una vez, podemos ejecutar en varias plataformas. En realidad la máquina virtual desempeña otras funciones, como la de aislar los programas Java al entorno de la máquina virtual, consiguiendo una gran seguridad.

Como herramienta se utiliza JBuilder 2.0, que es una herramienta RAD eficiente hasta cierto punto. Las máquinas utilizadas para el desarrollo son potentes y no tienen

problema, pero cuando se intenta utilizar en maquinas mas sencillas, los recursos que consume son importantes.

En el principio del proyecto se intento realizar la conexión entre cliente y servidor mediante métodos remotos (RMI) inmersos en un applet cliente y un servidor. Este método no funcionó, pues generó muchas restricciones en cuanto a seguridad. Los navegadores o browsers no permiten tan fácilmente una conexión de este estilo y si existen firewall y proxys entre las conexiones la comunicación es muy difícil. Por todo lo anterior se decidió cambiar a servlets, con lo cual la comunicación se realiza de forma conveniente y muy fácil.

3.1.5.- Técnicas de programación en HTML

Para que cualquier persona tenga un acceso fácil a la clase virtual así como información relativa con los cursos de la Universidad y acceso a información detallada y de uso gral., el recurso más sencillo es el uso de paginas HTML. Como un único requisito el usuario debe tener un navegador como Internet Explorer o Netscape. Al utilizar Java del lado servidor las mismas paginas pueden ser generadas de forma dinámica lo que hace muy rico su contenido y posibilidades. Las versiones actuales de HTML(Lenguaje de Marcas para HiperTexto) proporcionan la posibilidad de cargar imágenes animadas, crear ligas, botones con acciones predefinidas en un script (por ejemplo JavaScript) y otras posibilidades sin la necesidad de contar con una máquina virtual como en el caso de los applets, con lo que su uso es mucho más eficiente al no necesitar demasiados recursos y cargase de manera rápida. Sin embargo, como no almacenan información a través del tiempo ni tienen la capacidad de desplegar ventanas independientes (para un aviso por ejemplo) su uso se restringió al papel de interfaz donde fue posible.

Las técnicas de programación del lenguaje de HTML es relativamente sencillo, ya que no necesariamente se debe de hacer la construcción paso a paso, si no que es realmente con interfaces que facilitan el trabajo como los son el FrontPage u otros, estos mismos generan instantaneamente el código y las lineas como para ligar algunas paginas son relativamente sencillas.

3.1.6.-Tecnicas utilizadas en Bases de Datos (Access e Interbase).

Este es un manejador de bases de datos muy popular y sencillo de manejar. Su efectividad no es lo más conveniente para un trabajo de este estilo pero como su uso es extensivo y no tiene que hacerse un programa particular para la introducción de datos ni modificación de ellos, fue la elección más pertinente en su momento. Este problema se ha considerado como no trascendental porque la única ocasión en que se utiliza arduamente la base de datos es cuando los alumnos ingresan a la clase virtual. En todo el demás tiempo el maestro es el único que hace uso de ella para actualizar calificaciones o consultar historiales. Sin embargo, se hace notar que su eficiencia no es la adecuada, y al usarse JDBC en el lado servidor, se puede cambiar la BD con facilidad sin necesidad de alterar otra cosa que el driver odbc particular de la nueva base de datos en todos los servlets que la utilizan, en caso de tener problemas de desempeño.

4. Desarrollo práctico (Primer Prototipo).

En la etapa práctica del proyecto, se hizo el desarrollo, empleando las técnicas del Análisis y Diseño Orientado a Objetos (ADOO), en las que se aplicó el diseño de *Use Cases* (Casos de uso UC); así, para cada UC se analizó su modelo de requerimientos, que, de acuerdo con lo ya mencionado en el marco teórico del presente reporte, dicho modelo comprende los modelos de casos, de interfaz y del dominio del problema.

Es importante mencionar, que hasta esta etapa el proyecto consta de dos prototipos, en esta parte explicaremos el desarrollo del primer prototipo para finalmente pasar a la descripción del segundo prototipo.

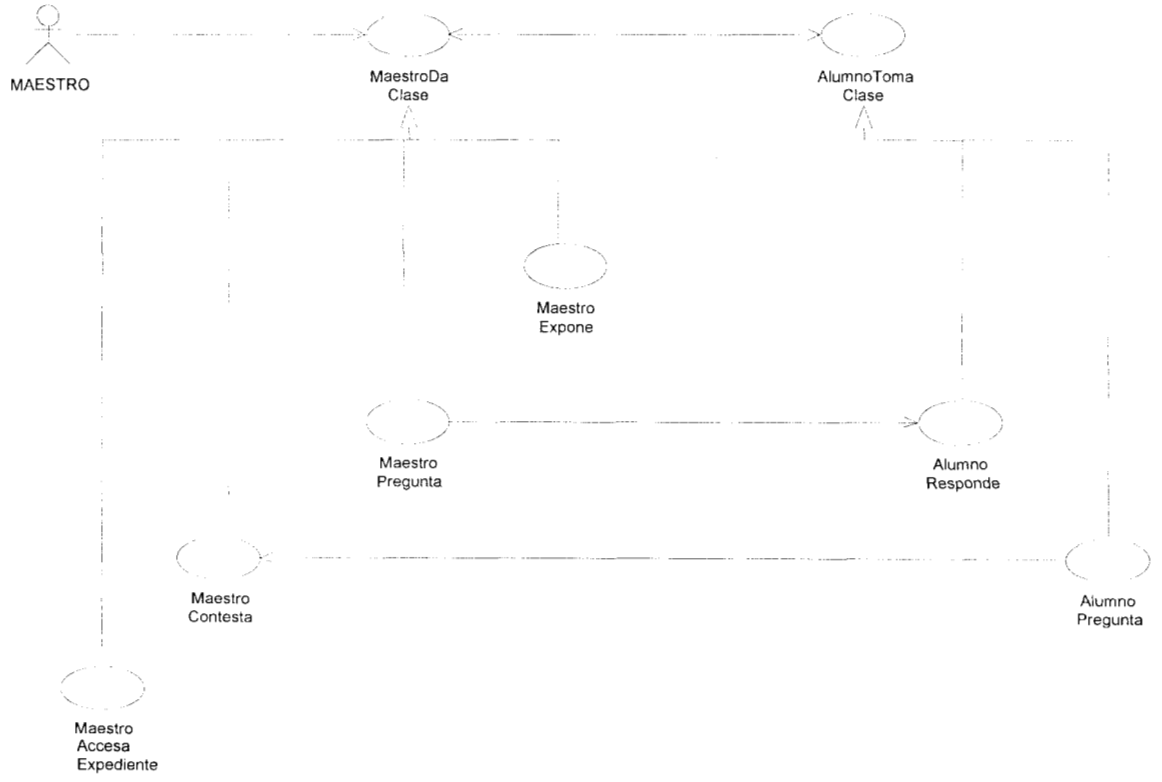
4.1 Modelo de requerimientos a nivel general.

En ésta etapa, se estableció la forma de trabajo el equipo y se comenzó con la familiarización del equipo de desarrollo con el problema (el diagrama se encuentra en el ANEXO 1).

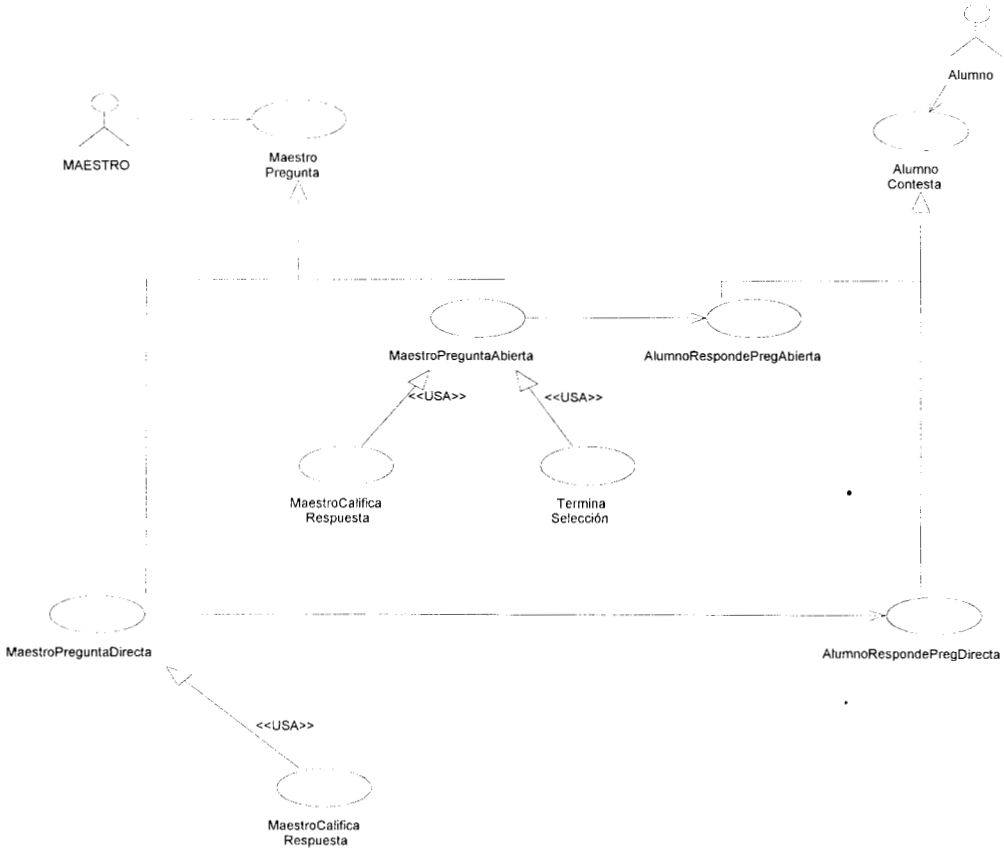
ANEXO 1

MODELO DE REQUERIMIENTOS (nivel general)

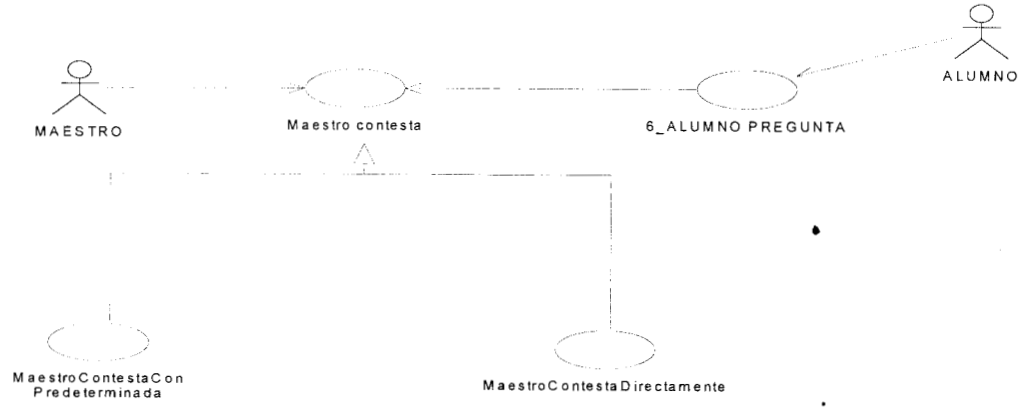
Diagrama General de Use Cases del Chat



Especialización de Use Case Maestro Pregunta y Alumno Responde



Especialización de Use Case Maestro Contesta



Especialización de Use Case Maestro Expone

4.2 Modelo de requerimientos para cada Use Case.

Aquí se especifica la secuencia de acciones a tomar por el sistema para cada intercambio de información (Ver diagramas en ANEXO 2).

4.2.1 Modelo de requerimientos:

4.2.1.1 Modelo de casos (Ver ANEXO2).

4.2.1.2 Modelo de Interfaz (Ver ANEXO2).

4.2.1.3 Modelo del dominio del problema (Ver ANEXO2).

4.2.2 Modelo de Análisis.

4.2.2.1 Escenarios explorados.

Los escenarios que se exploraron, corresponden únicamente al encontrado en el marco de "Simple Correcto", ya que al ingresar a otro escenario, en el que la clave de acceso del alumno o el profesor sean incorrectas, el sistema simplemente no entra en ejecución, por lo que se omitieron diagramas que no tiene gran valor ilustrativo.

4.2.3 Modelo de Diseño (Diagramas de Secuencia).

4.2.3.1 Escenarios Explorados.

Como se estableció en el modelo de análisis, el único escenario descrito es el que presenta las características de ser el "Simple Correcto", por lo que se presentan los diagramas de secuencia para cada Use Case para éste escenario en particular (Ver ANEXO3).

ANEXO 2:
Modelos de Requerimientos para cada UC

1. Use Case MaestroPreguntaAbierta

PASOS

1.-Maestro presiona -pregunta abierta
2.-Teclea pregunta en el área de texto.
3.-Maestro presiona enviar.
4.-Sistema lleva pregunta a todos los alumnos.
5.-Sistema trae la solicitud de los alumnos que quieren responder y los marca en el área de alumnos presentes.
6.-Maestro selecciona alumno que responderá y presiona -Aceptar.
7.-Sistema avisa a alumno que ha sido seleccionado para responder, comienza un timer y trae su respuesta al área de texto, difundiendo ésta a todos los demás clientes.
8.-U.C. MaestroCalificaRespuesta y regresa a 6 o pasa a 9.
9.-U.C. TerminaSelección.

Diagrama de Estado (Use Case Maestro Pregunta Abierta)

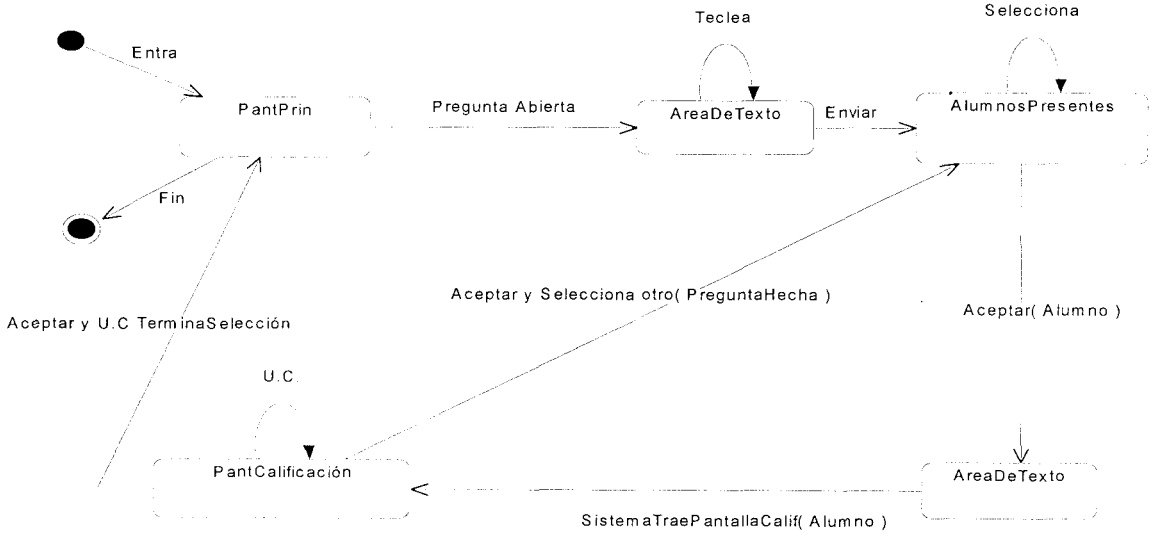
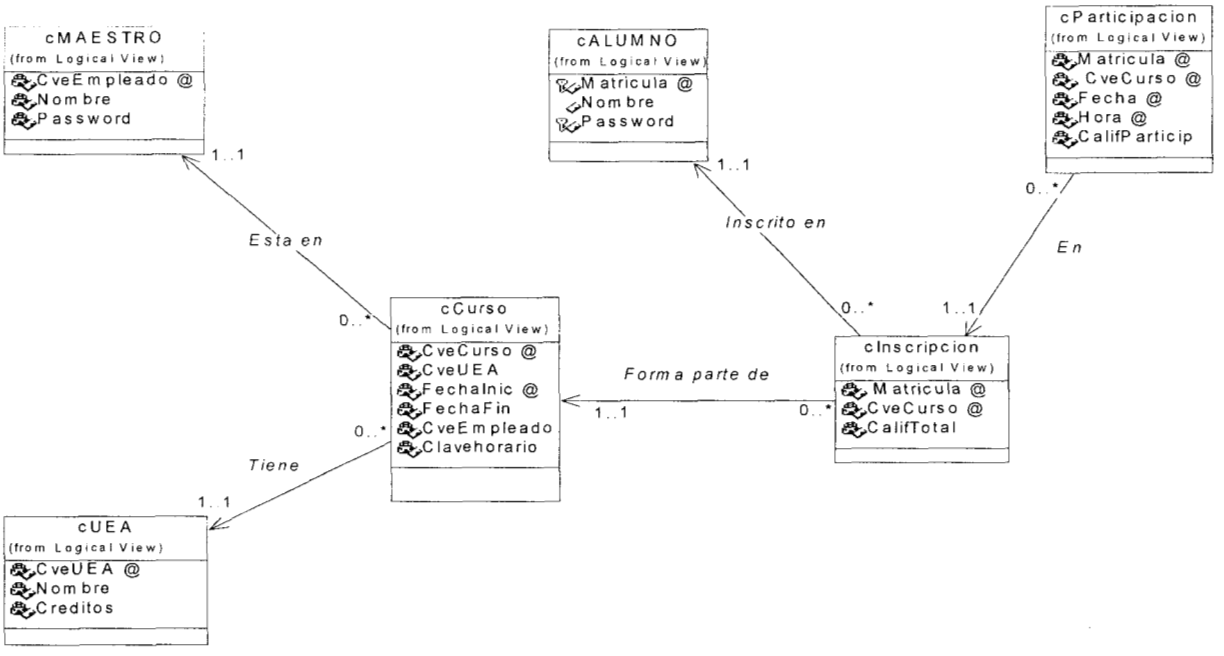


Diagrama de Clases (Use Case Maestro Pregunta Abierta)

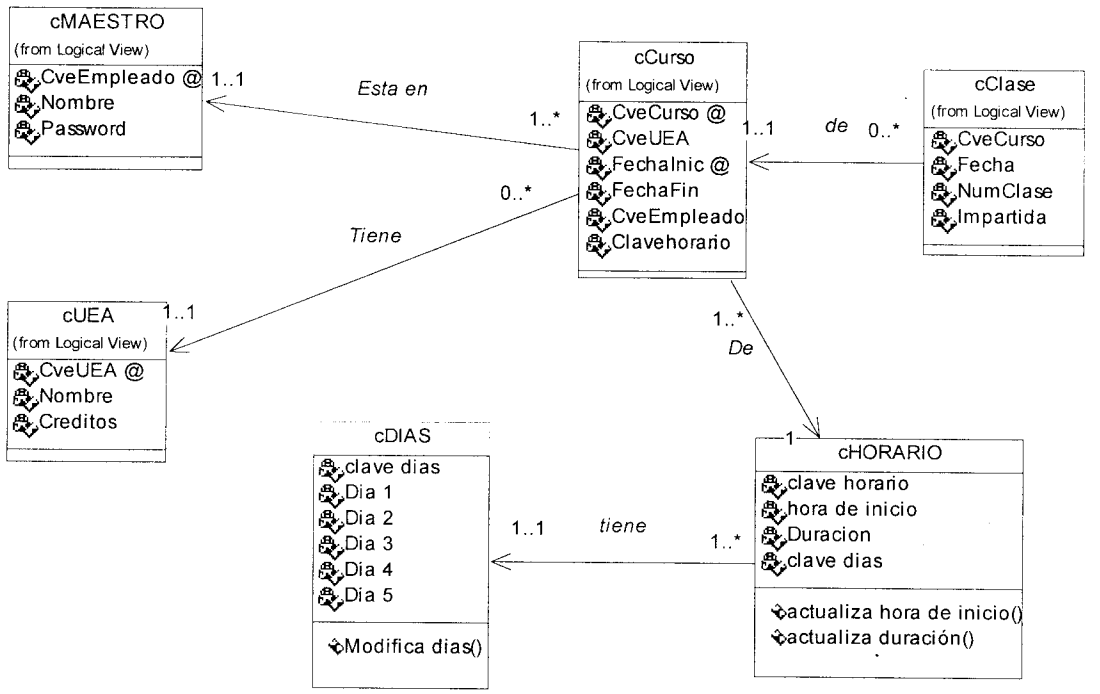


2. Use Case MaestroCalificaRespuesta

PASOS

Maestro otorga una calificación a la respuesta o comentario del alumno.
1.-Sistema trae un cuadro de dialogo con los datos del alumno previamente seleccionado y un área de texto para que el maestro ponga la calificación.
2.-Maestro teclea la calificación y presiona aceptar.
3.-Sistema guarda calificación del alumno.
4.-Sistema muestra al alumno seleccionado mediante un cuadro de dialogo el aviso de que su respuesta fue calificada y la calificación correspondiente.
5.-Sistema despliega en todos los clientes de alumnos la participación exitosa y calificación del alumno en el área de texto.

Diagrama de Clases (Use Case MaestroCalificaRespuesta)



3. Use Case Termina Selección pregunta Abierta

PASOS
1.-Maestro presiona cancelar.
2.-Sistema borra todas las marcas de los aspirantes a contestar.
3.-Sistema avisa a todos los alumnos que se ha terminado la oportunidad de contestar.

Modelo de Interfaz

Diagrama de Estado (Use Case Termina Selección Pta. Abierta)

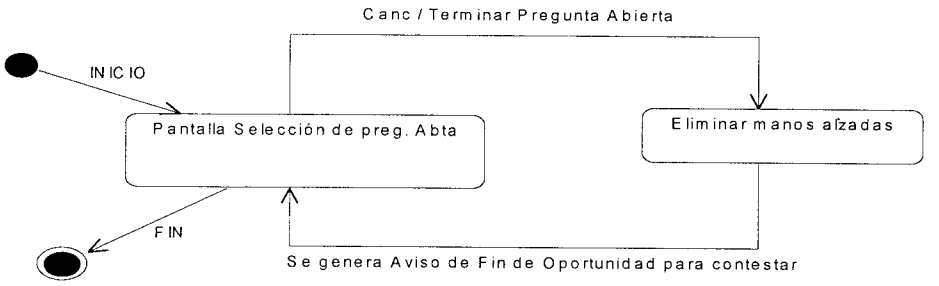
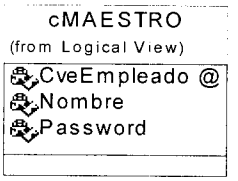


Diagrama de Clases (Use Case Termina Selección Pta. Abierta)



4. Use Case MaestroPreguntaDirecta

PASOS

Maestro realiza una pregunta específica para un alumno y la califica.

1.-Maestro selecciona alumno en la pantalla de alumnos presentes.
2.-Maestro presiona pregunta directa.
3.-Maestro teclea la pregunta en el área de texto y presiona enviar.
4.-Sistema lleva pregunta a todos los alumnos.
5.-Sistema avisa a alumno mediante cuadro de dialogo que ha sido seleccionado para contestar y comienza un timer..

6.-Sistema trae respuesta y la despliega en todos los clientes.
7.-U.C. MaestroCalificaRespuesta.
8.-Sistema termina.

Modelo de Interfaz

Diagrama de Estado (Use Case MaestroPreguntaDirecta)

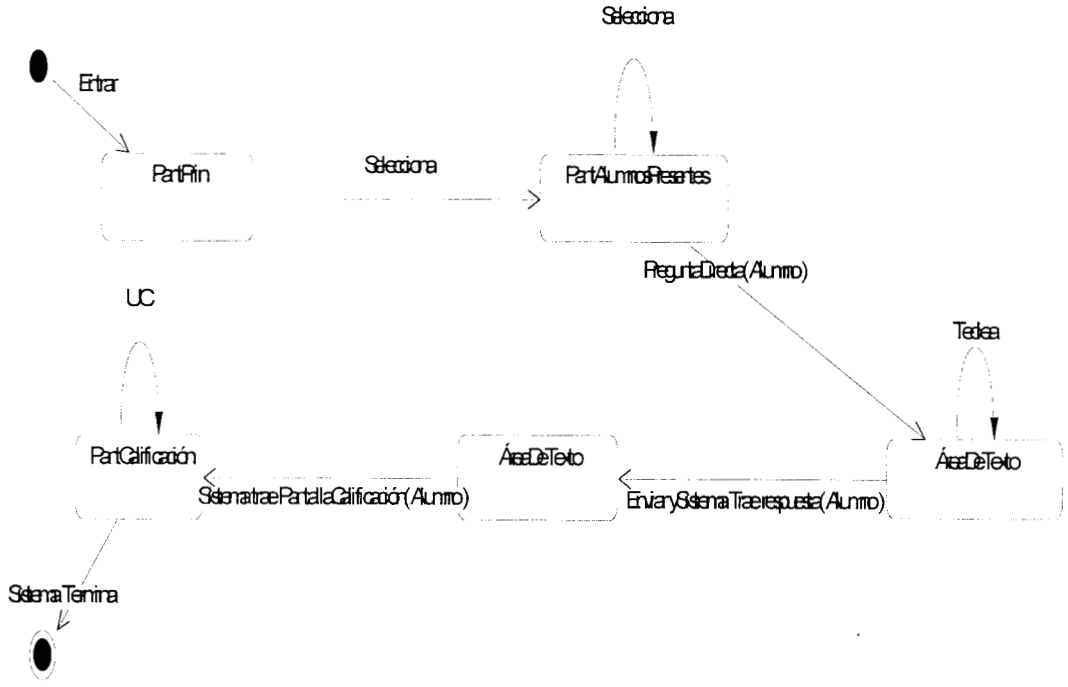
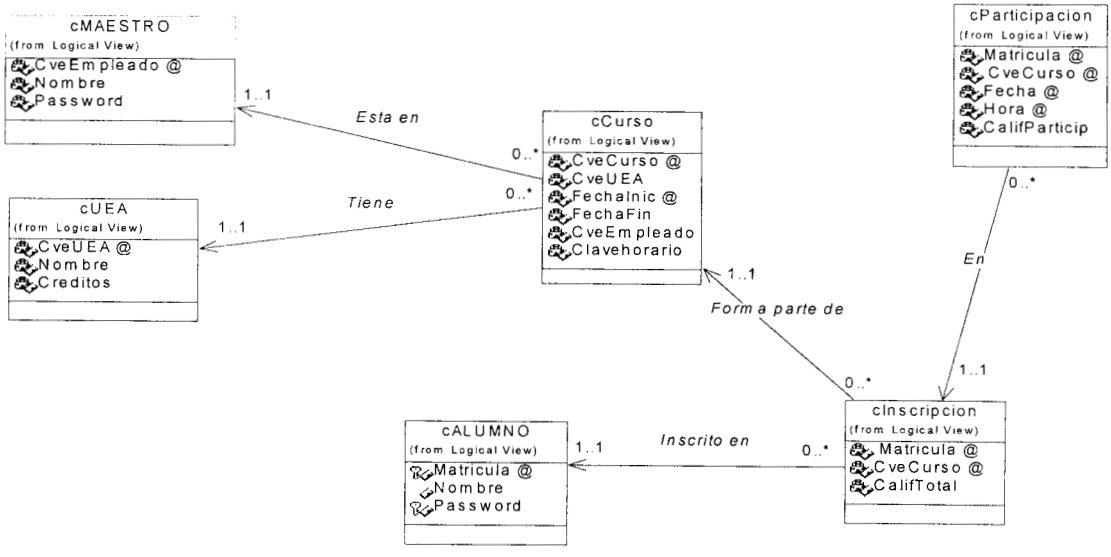


Diagrama de Clases (Use Case MaestroPreguntaDirecta)



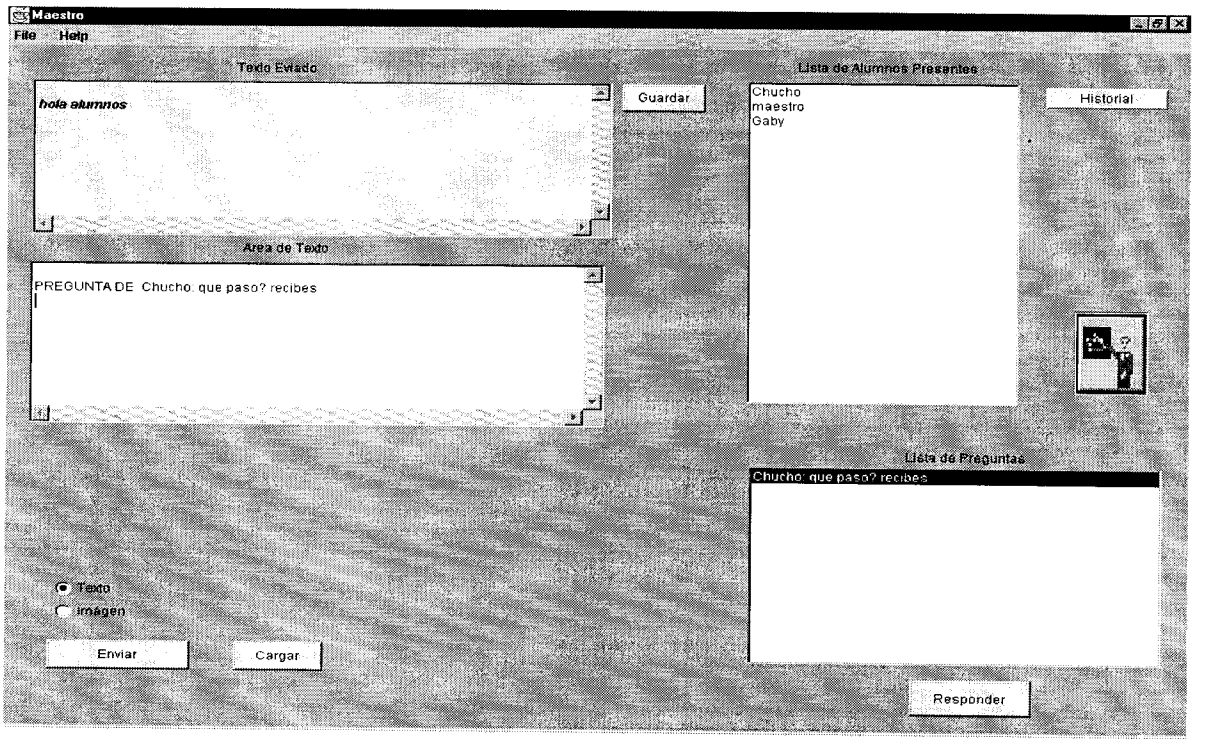
5. Use Case MaestroContestaDirectamente

Ante las diversas preguntas que el maestro puede ver en el área de preguntas que realizan los alumnos, selecciona una para responder y la contesta .

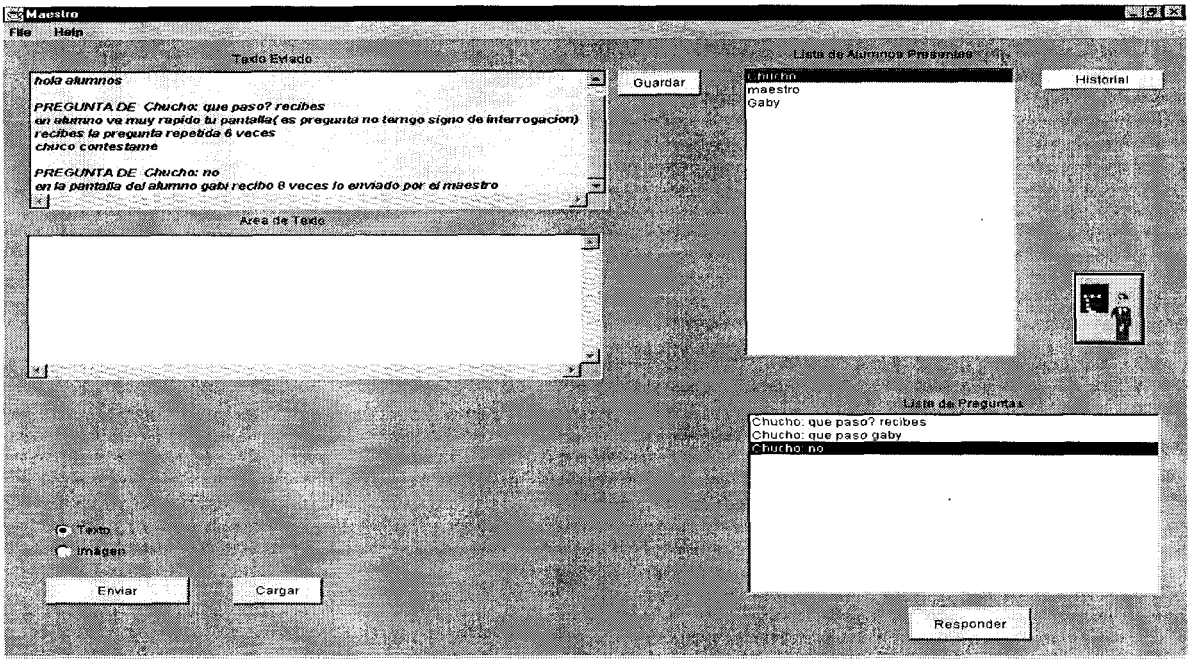
PASOS

1.-Maestro selecciona la pregunta correspondiente en el área de preguntas.
2.-Maestro presiona responder.
3.-Sistema prepara envío.(Pregunta, alumno).
4.-Maestro teclea respuesta en el área de texto y presiona enviar
5.-.Sistema despliega en todos los alumnos la pregunta , el alumno que la realizó y la respuesta dada por el Maestro.

Layouts de pantallas(Use Case MaestroContestaDirectamente)



1.-Maestro selecciona la pregunta correspondiente en el área de preguntas.



2.-Maestro teclea respuesta en el área de texto y presiona enviar

Diagrama de Estado (Use Case MaestroContestaDirectamente)

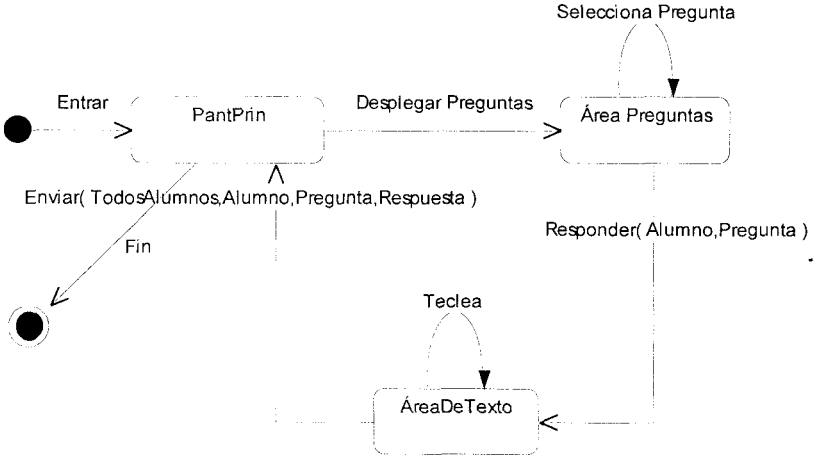
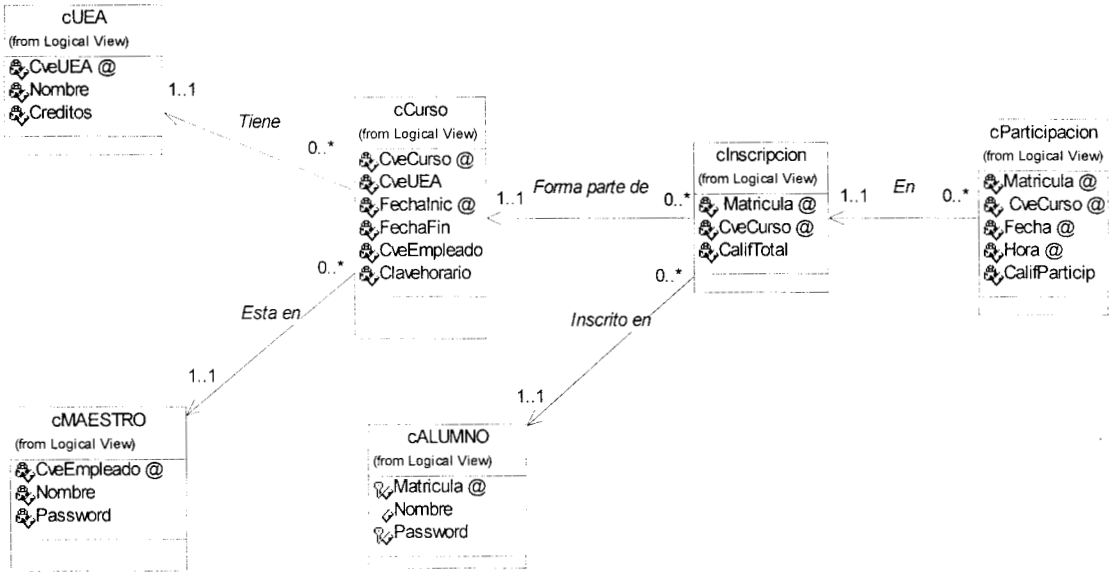
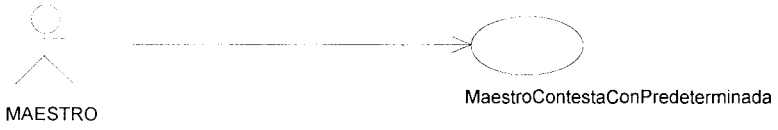


Diagrama de Clases (Use Case MaestroContestaDirectamente)



6. Use Case MaestroContestaConPredeterminada



Maestro contesta una pregunta de las que se han hecho en el área de preguntas y la contesta mediante una respuesta que previamente ha capturado y puesto a disposición.

PASOS

1.-Maestro selecciona la pregunta que desea responder en el área de preguntas.
2.-Maestro presiona -Responder.
3.-Sistema prepara respuesta(Pregunta, alumno).
4.-Maestro presiona -Cargar.
5.-Sistema trae todas las opciones que se han precargado en un cuadro de opciones que contiene la pregunta y la respuesta.
6.-Maestro selecciona la pregunta y respuesta correspondiente y presiona -Aceptar.
7.-Sistema despliega en el área de texto la pregunta y respuesta seleccionada.
8.-Maestro presiona -Enviar.
9.-Sistema lleva la pregunta, el alumno que la realizó y la respuesta dada a todos los alumnos de la clase.

Diagrama de Estado (Use Case MaestroContestaConPredeterminada)

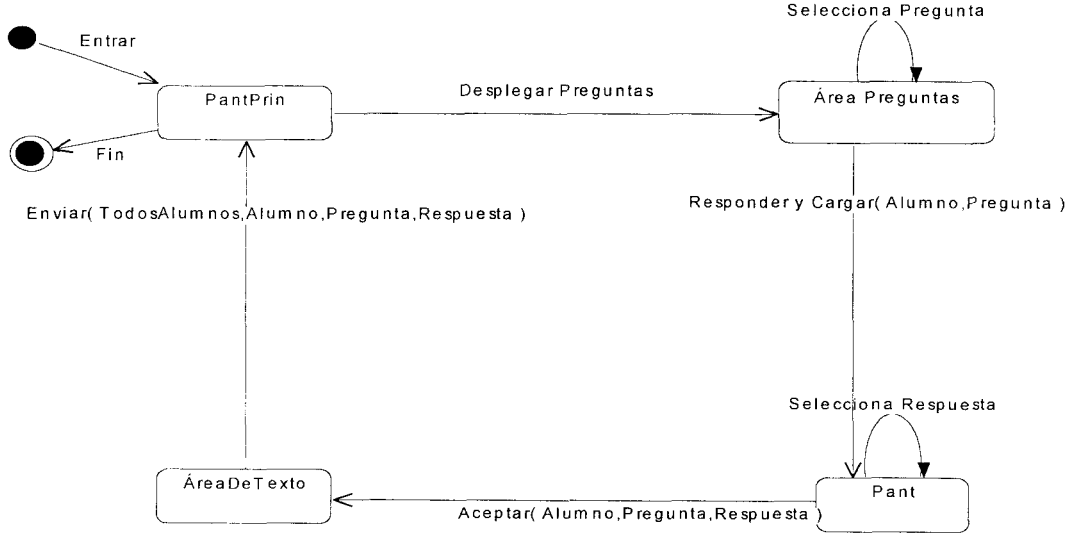
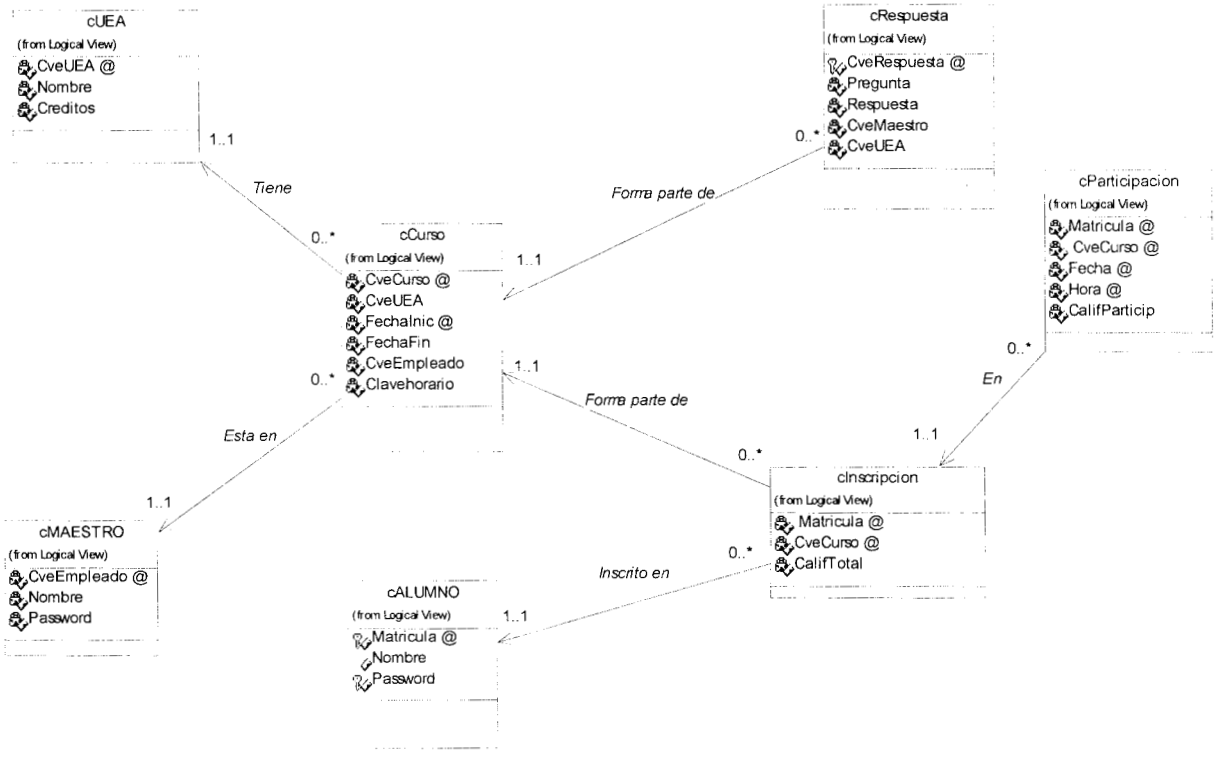


Diagrama de Clases (Use Case MaestroContestaConPredeterminada)



7. Use Case AlumnoRespondePregDirecta

PASOS

1.-Sistema avisa que el alumno ha sido seleccionado para responder la pregunta directa .Despliega un timer(reloj).
2.-Alumno teclea respuesta y presiona enviar.

3.-Sistema lleva respuesta a servidor o avisa del fin del tiempo.(para contestar).
4.-Sistema despliega mensaje de respuesta recibida o puntuación otorgada.(privada).

Diagrama de Estado (Use Case AlumnoRespondePregDirecta)

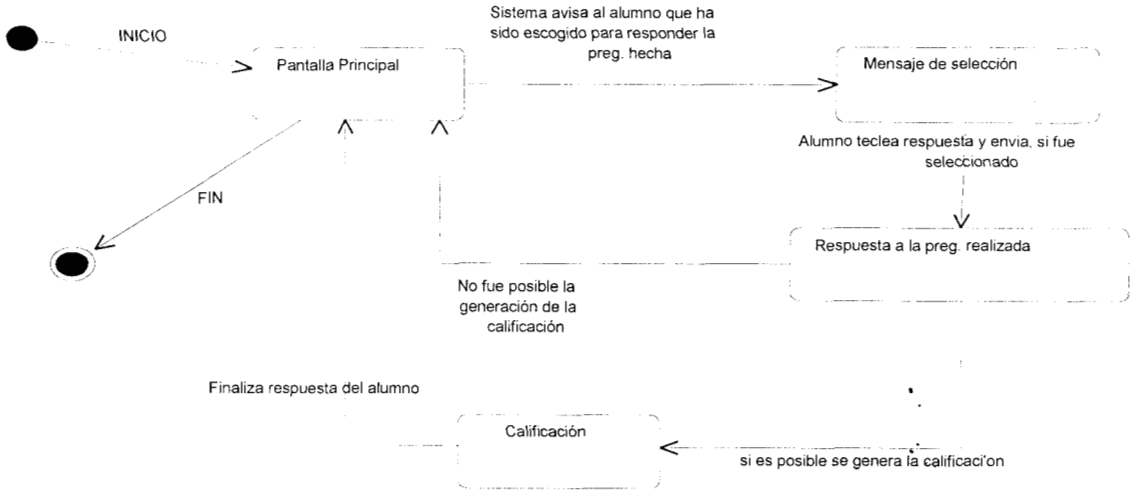
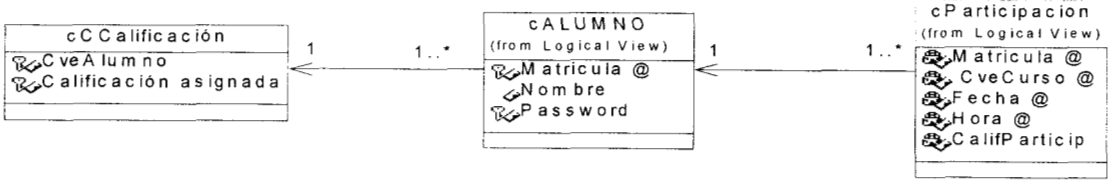


Diagrama de Clases (Use Case AlumnoRespondePregDirecta)



8. Use Case AlumnoRespondePregAbierta

PASOS

1.-Sistema trae a la pantalla la pregunta y la marca como abierta(Area de Exposición).
2.-Alumno presiona solicitud de Participación .
3.-Sistema lleva solicitud al servidor.
4.- Sistema trae respuesta de servidor
5.-Alumno teclea respuesta y presiona enviar.
6.-Sistema lleva respuesta del alumno.
7.-Sistema notifica a los alumnos del termino de la sección de Respuestas.

Layouts de pantallas (Use Case AlumnoRespondePregAbierta)

Diagrama de Estado (Use Case AlumnoRespondePregAbierta)

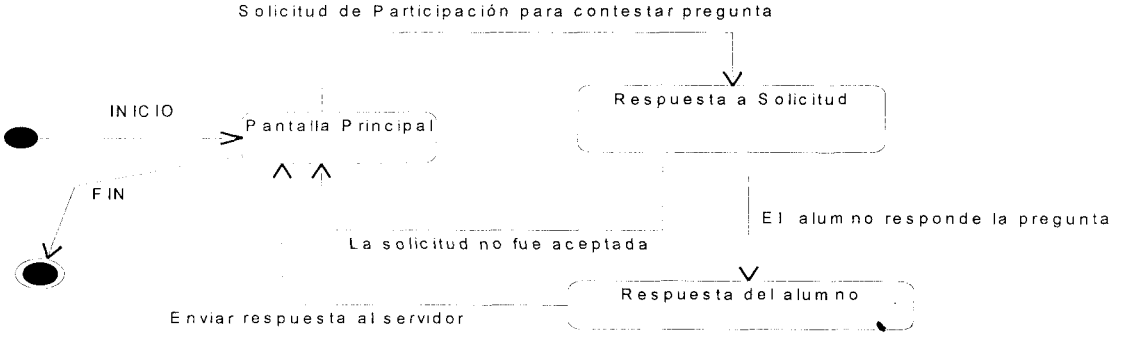
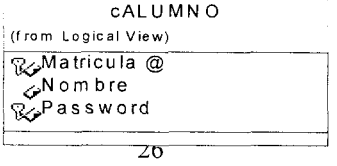
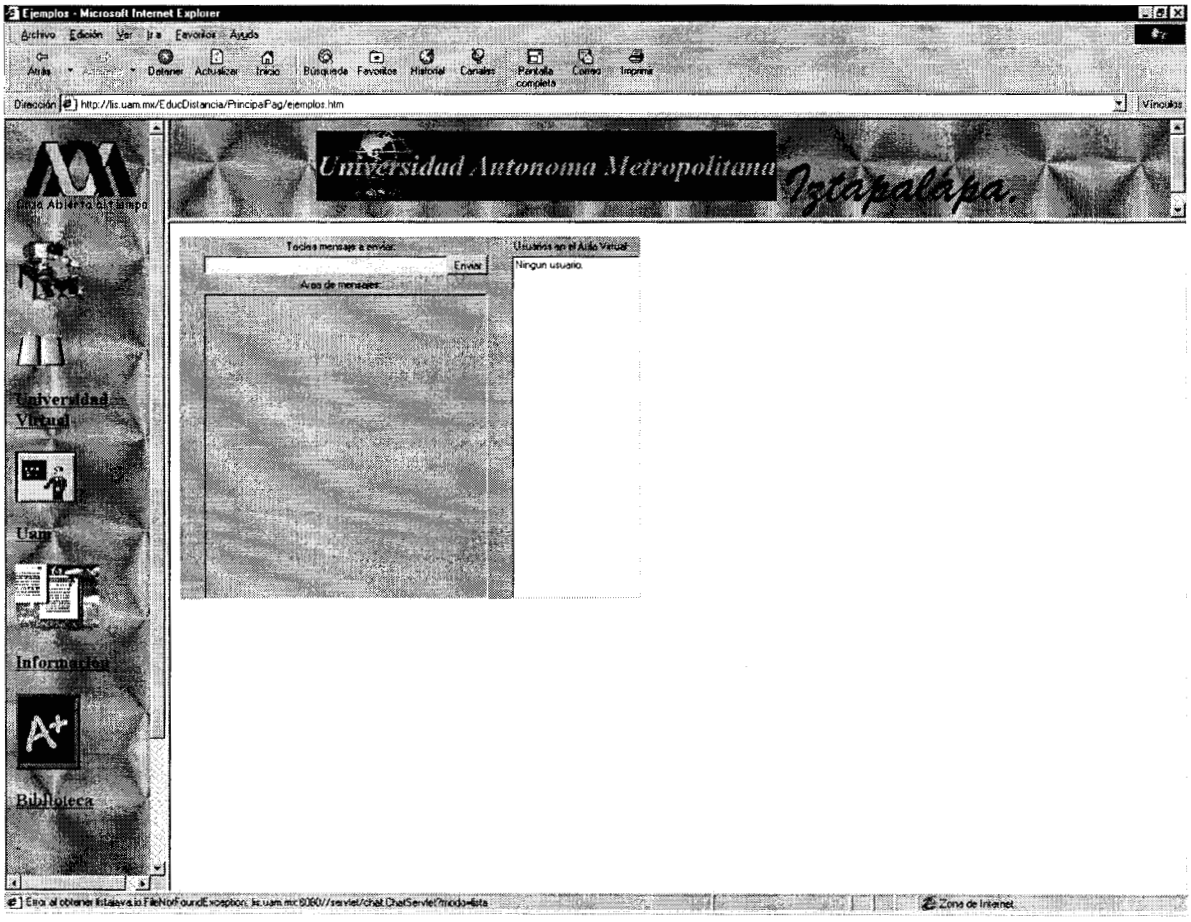


Diagrama de Clases (Use Case AlumnoRespondePregAbierta)



Layouts de pantallas:
AlumnoRespondePreguntaAbierta



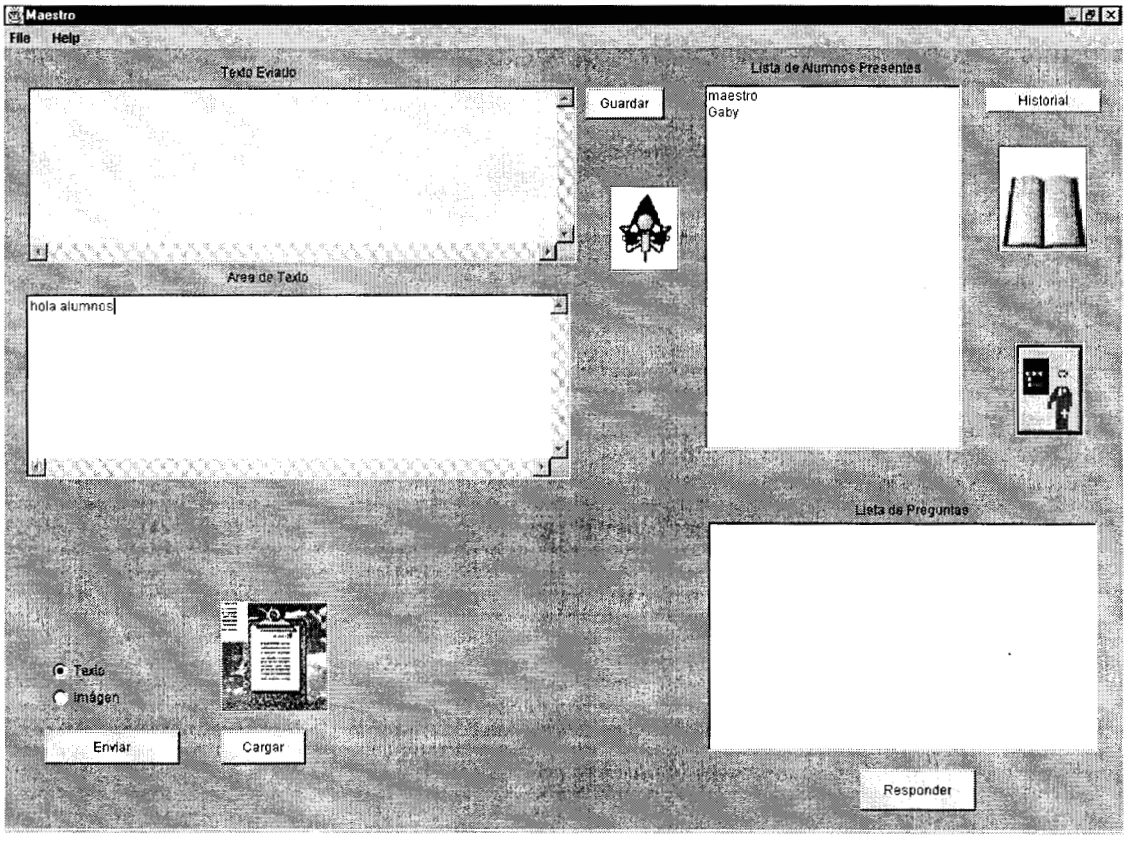
9. Use Case MAESTRO EXPONE DIRECTO

PASOS

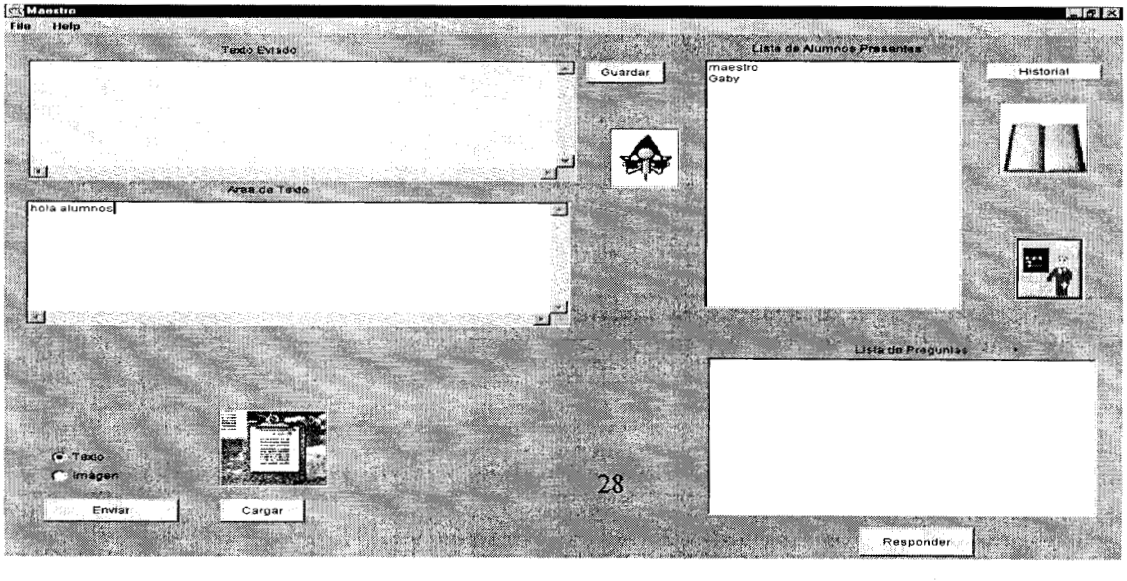
1.- Maestro elige expone directo

2.- Sistema muestra pantalla del maestro.
3.- Maestro teclea en el área de pizarrón.
4.- Sistema envía información a alumnos
5.- Maestro termina clase.

Layouts de pantallas(Use Case MAESTRO EXPONE DIRECTO)

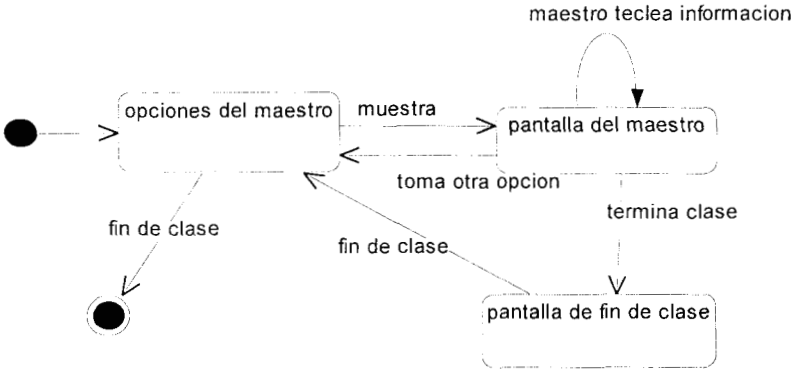


3.- Maestro teclea en el área de pizarrón.



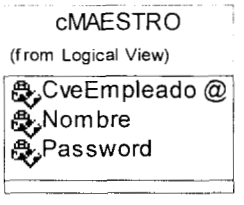
4.- Sistema envía información a alumnos

Diagrama de Estado (Use Case MAESTRO EXPONE DIRECTO)



Modelo del Dominio del problema

Diagrama de Clases(Use Case MAESTRO EXPONE DIRECTO)

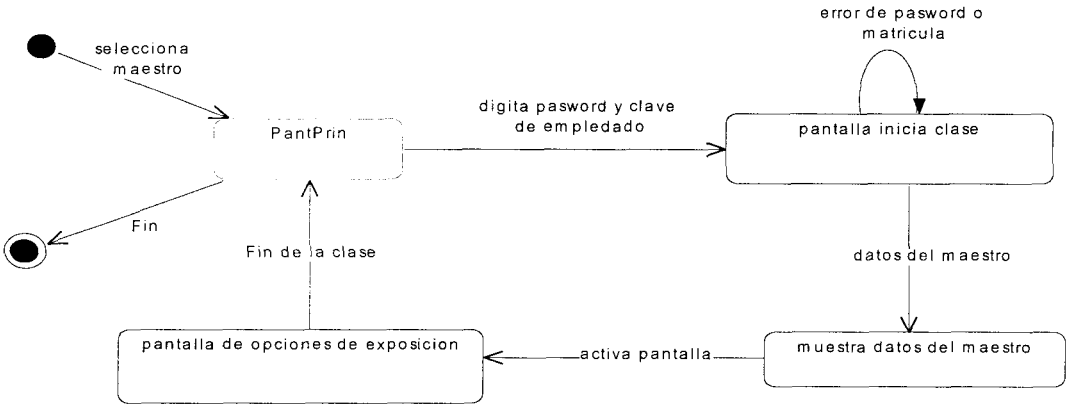


10. Use Case MaestroIniciaSesión

PASOS

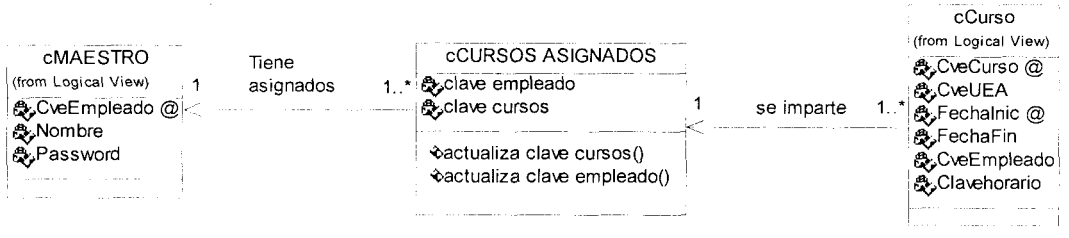
1.- MAESTRO TECLEA PASWORD.
2.- MAESTRO TECLEA CLAVE DE EMPLEADO.
3.- SISTEMA MUESTRA PANTALLA DE ELECCION DE EXPOSICION.
4.- MAESTRO ELIGE TIPO DE EXPOSICION.
6.- SISTEMA MUESTRA PANTALLA DEL MAESTRO.
7.- U.C. MAESTRO DA LA CLASE.
8.- MAESTRO TERMINA SESION
10.-SISTEMA VERIFICA LA HORA DE TERMINACION
11.- SISTEMA AVISA LA TERMINACION DE LA CLASE
12.- SISTEMA CIERRA LA CLASE

Diagrama de Estado (Use Case MaestroIniciaSesión)



Modelo del Dominio del problema

Diagrama de Clases (Use Case MaestroIniciaSesión)



11. Use Case AlumnoEntra a clase

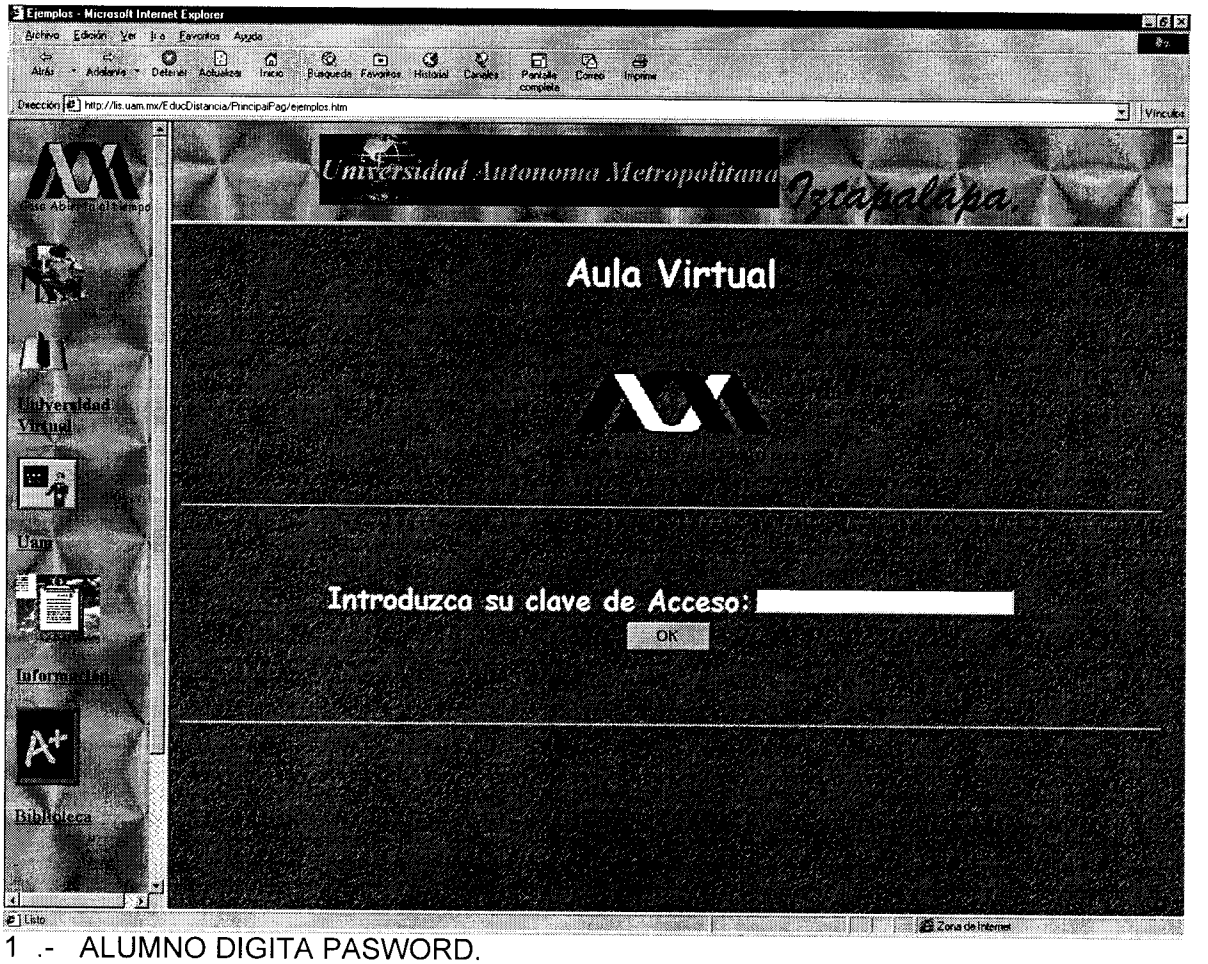
PASOS

1.- ALUMNO DIGITA PASWORD.

2.- ALUMNO DIGITA MATRICULA.
3,- SISTEMA VERIFICA LA HORA Y CLASE QUE PUEDE TOMAR A ESA HORA.
4.- SISTEMA MUESTRA LA PANTALLA DEL CURSO.
5.- ALUMNO TECLEA DE ACEPTACION.
6.- SISTEMA MUESTRA PANTALLA DEL ALUMNO.
7.- U.C.ALUMNO TOMA CLASE.
8.- SISTEMA TERMINA SECION.
9.- ALUMNO TERMINA CLASE
10.- ALUMNO VISUALIZA PUNTUACION AL FINALIZAR LA CLASE.
11.- SISTEMA REGRESA A LA PANTALLA PRINCIPAL

Modelo de Interfaz:

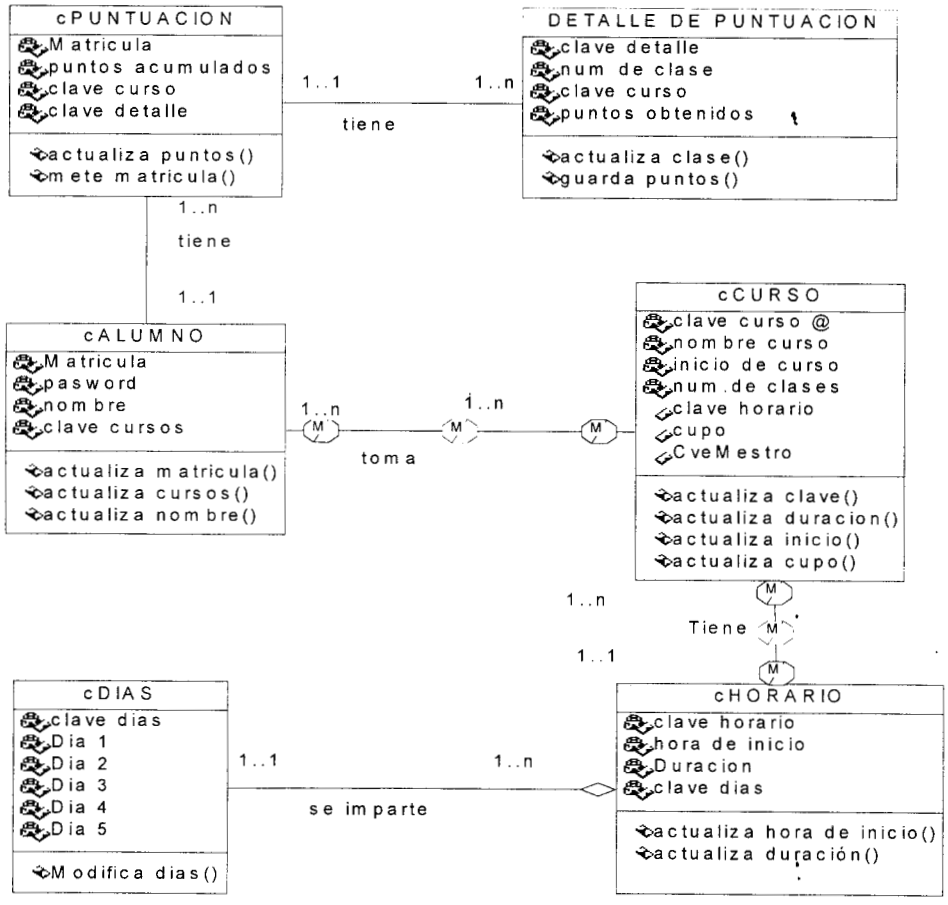
Layouts de pantallas



1 .- ALUMNO DIGITA PASWORD.

Modelo del Dominio del problema

Diagrama de Clases (Use Case AlumnoEntra a clase)



12. Use Case MaestroAccesaExpediente

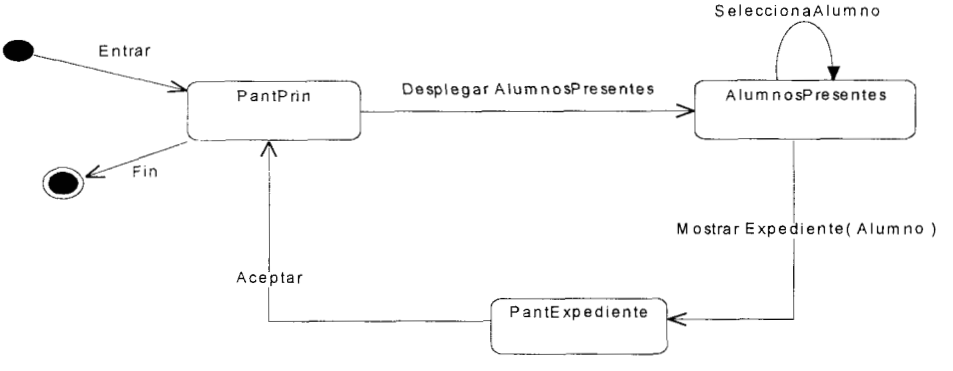
PASOS

Maestro accesa a el historial académico de un alumno en particular.

1.-Maestro selecciona alumno de la pantalla de alumnos presentes.
2.-Maestro presiona Historial.
3.-Sistema muestra el historial del Alumno y lo despliega en una pantalla especial.
4.-El Maestro presiona el botón mostrar foto de la pantalla especial.
5.-El sistema despliega la foto del Alumno en la pantalla especial.
6.-El Maestro presiona aceptar.
7.-El sistema deshabilita la pantalla de historial y regresa a la pantalla principal.

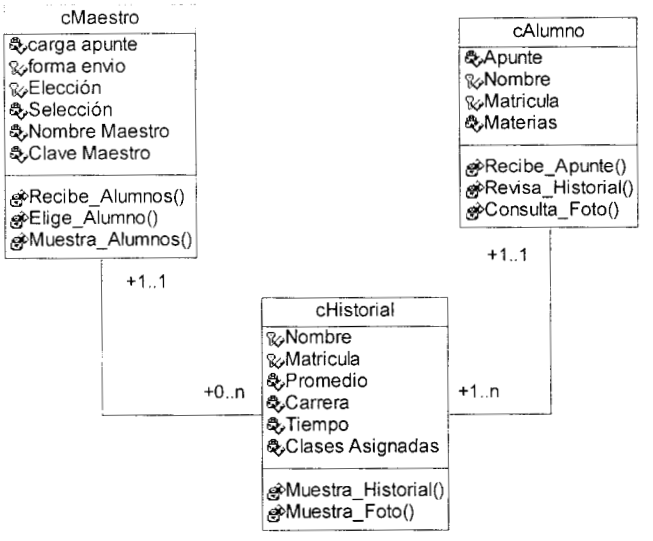
Modelo de Interfaz

Diagrama de Estado (Use Case MaestroAccesaExpediente)

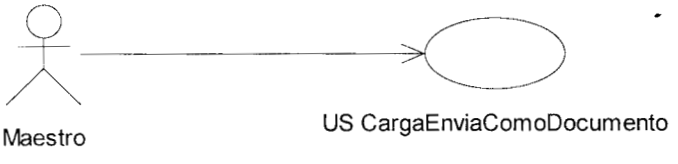


Modelo del Dominio del problema

Diagrama de Clases (Use Case MaestroAccesaExpediente)



13. Use Case CargaEnviaComoDocumento



PASOS:

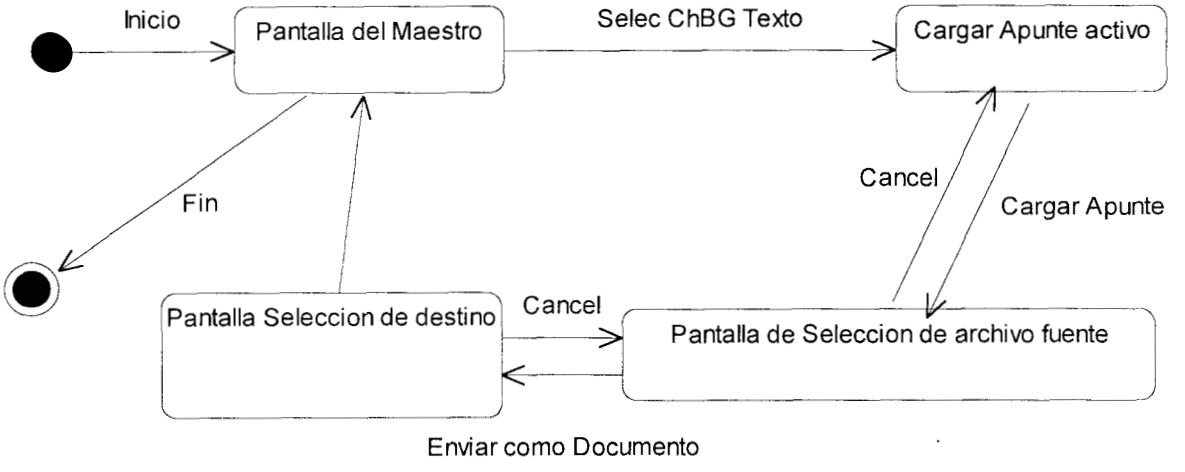
El maestro selecciona enviar un texto como Documento

1. Maestro selcciona del CheckBoxGroup 'texto'
2. El sistema activa boton 'Carga Apunte'
3. El maestro presiona botón 'Carga Apunte'
4. El sistema despliega pantalla de opción 'Enviar como documento'
5. El sistema abre pantalla de selección del destino del archivo
6. El sistema despliga pabntalla de selección de archivo
7. El maestro busca selecciona archivo con texto
8. El maestro presiona 'Abrir en la pantalla de selección
9. El sistema cierra pantallas de selccion
10. El sistema envia el documento al destino indicado

11. El maestro continúa con la clase

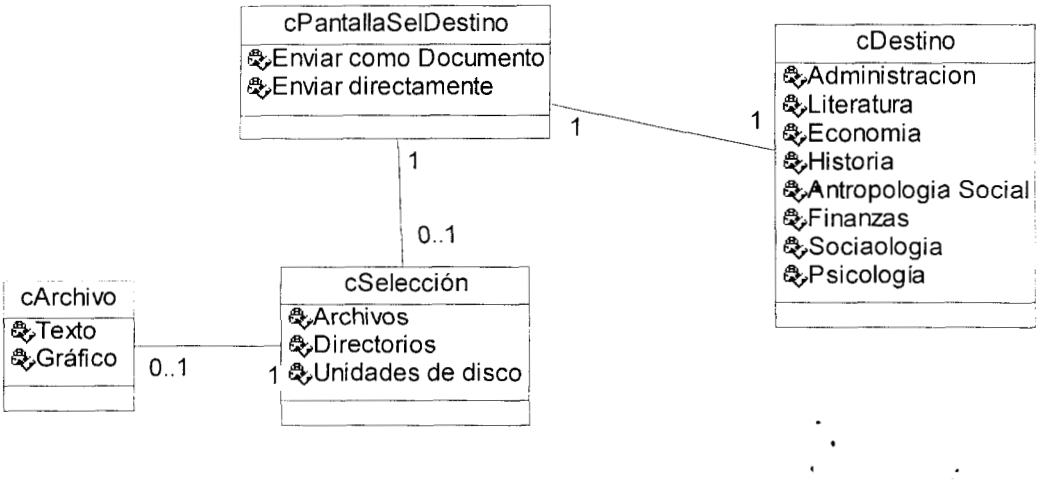
Modelo de Interfaz

Diagrama de estados UC CargaEnviaComoDocumento



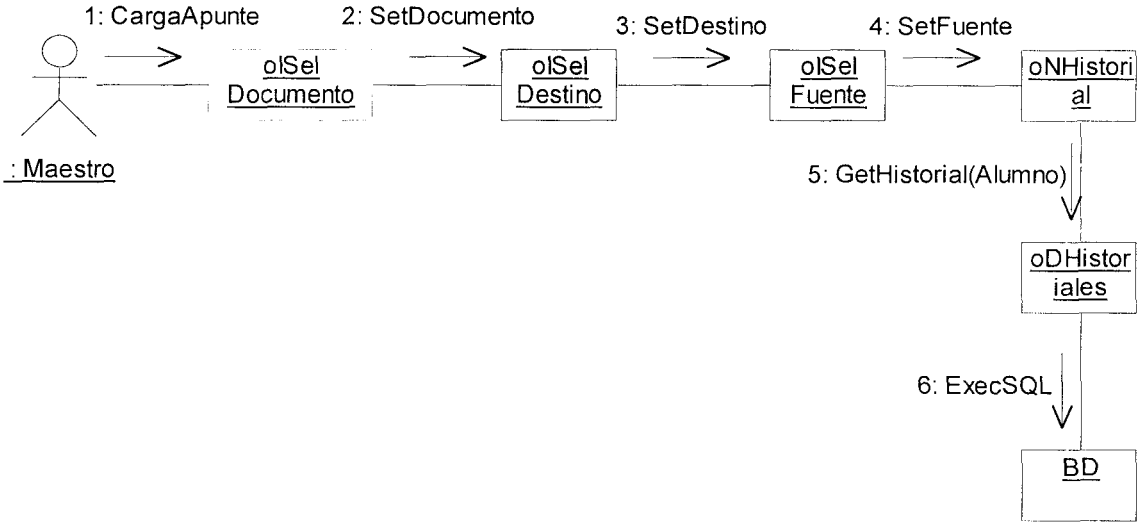
Modelo del Dominio del Problema

UC CargaEnviaComoDocumento



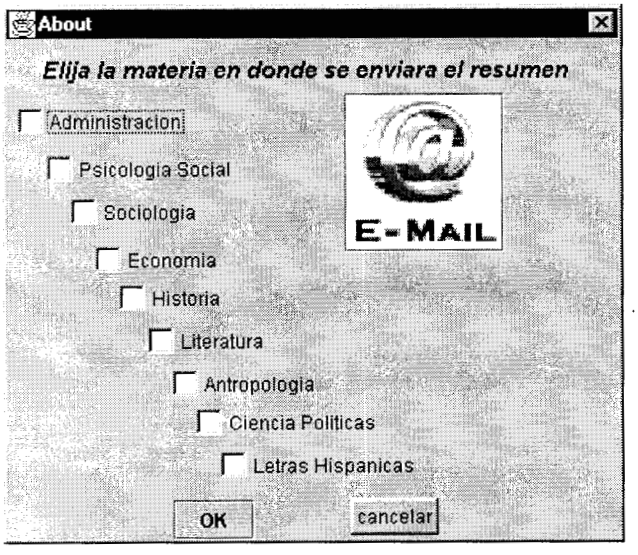
Modelo de Colaboración

UC CargaEnviaComoDocumento



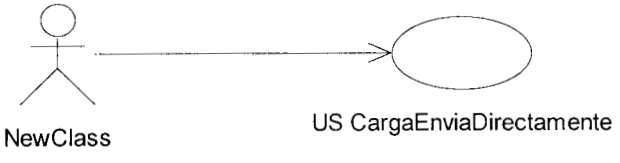
Layouts de pantallas

1. Pantalla de selección de tipo de envio



2. Pantalla de selección de destino de envío

14. UC CargaEnviaDirectamente

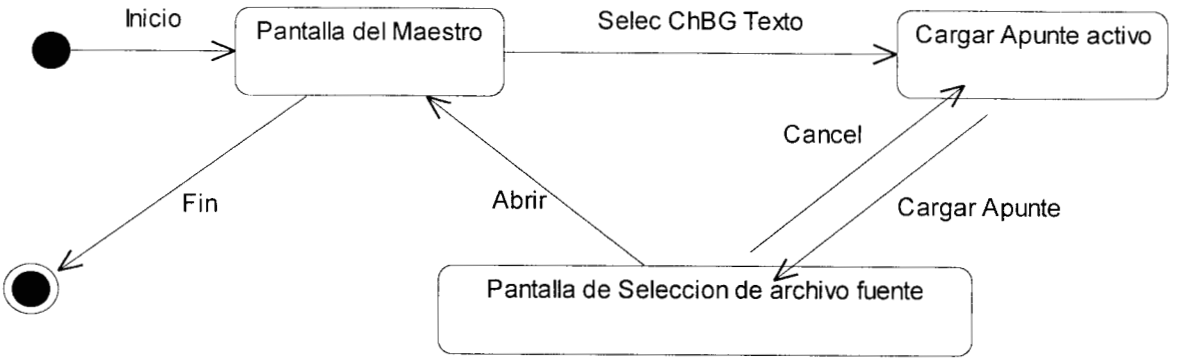


PASOS

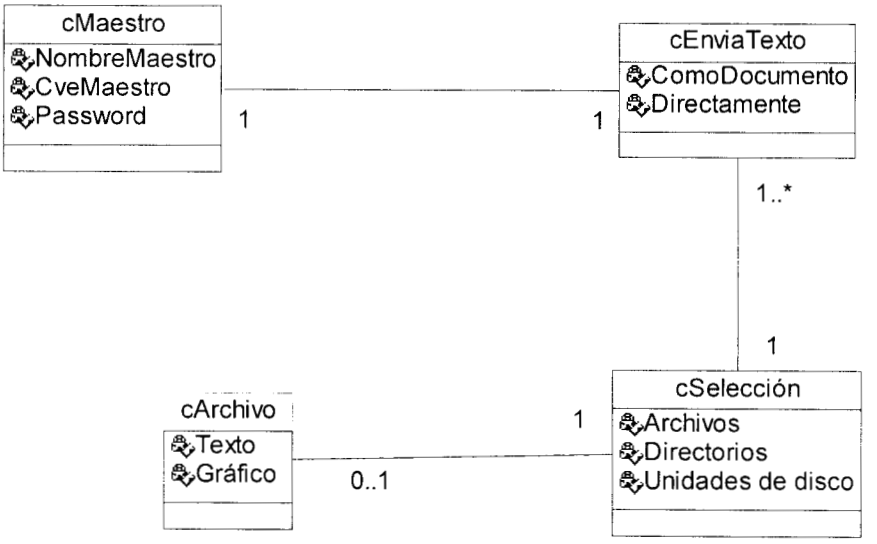
1. Maestro selcciona del CheckBoxGroup 'texto'
2. El sistema activa boton 'Carga Apunte'
3. El maestro presiona botón 'Carga Apunte'
4. El sistema despliega pantalla con opción 'Enviar Directamente'
5. El Sistema despliega pantalla de selccion de archivo
6. El maestro busca y selecciona archivo con texto a enviar
7. El sistema cierra pantalla de seleccion
8. El sistema despliega el documento en el area de texto
9. El maestro continúa con la clase

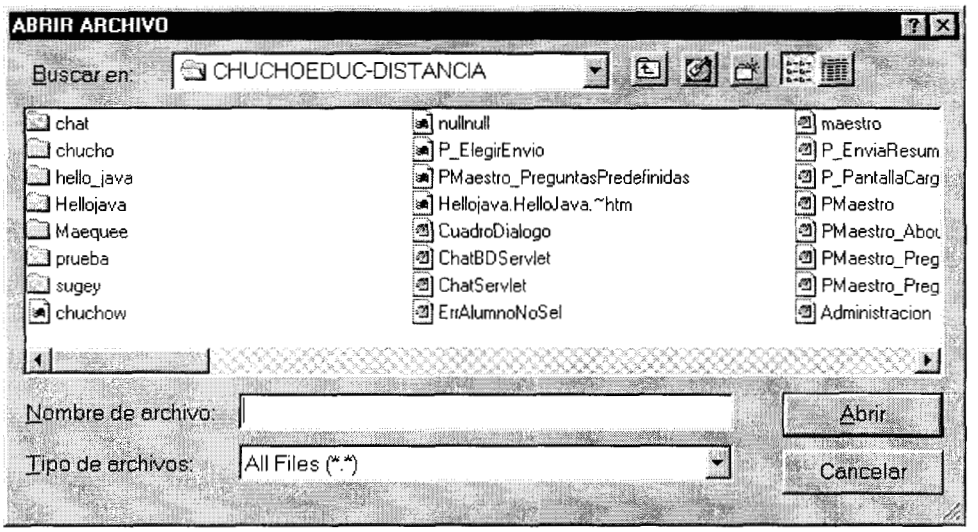
Modelo de Interfaz

Diagrama de Estado UC CargaEnviaDirectamente



Modelo del Dominio del Problema
UC CargaEnviaDirectamente

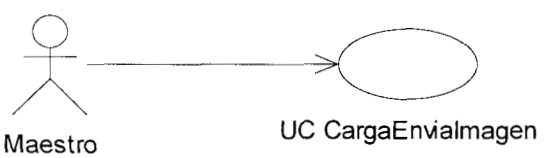




2. Pantalla de selección de archivo a enviar

15. USE CASE CARGAENVIAIMAGEN.

PASOS



(EN LA PANTALLA DEL MAESTRO, SELECCIONAN UN ICONO)

1. Maestro selecciona del checkBoxGroup 'Imagen'
2. El sistema activa botón 'Carga Imagen'
3. El maestro presiona botón 'Carga Imagen'
4. El sistema despliega pantalla 'CImagenes'
5. Maestro presiona 'Abrir Imagen'
6. El sistema despliega pantalla de selección de archivos
7. El maestro busca y selecciona archivo con Imagen (*.gif, *.jpg, *.jpeg, *.bmp, etc...)
8. Maestro presiona botón 'Abrir' de la pantalla de selección
9. El sistema muestra la imagen en un recuadro
10. Maestro presiona botón 'Enviar Imagen'
11. Sistema desaparece pantalla Cimagenes
12. El sistema coloca la imagen en el recuadro del pizarrón
13 El maestro continua la clase en la pantalla de maestro

Modelo de Interfaz
Diagrama de Estado(Use Case CargaEnvialmagen)

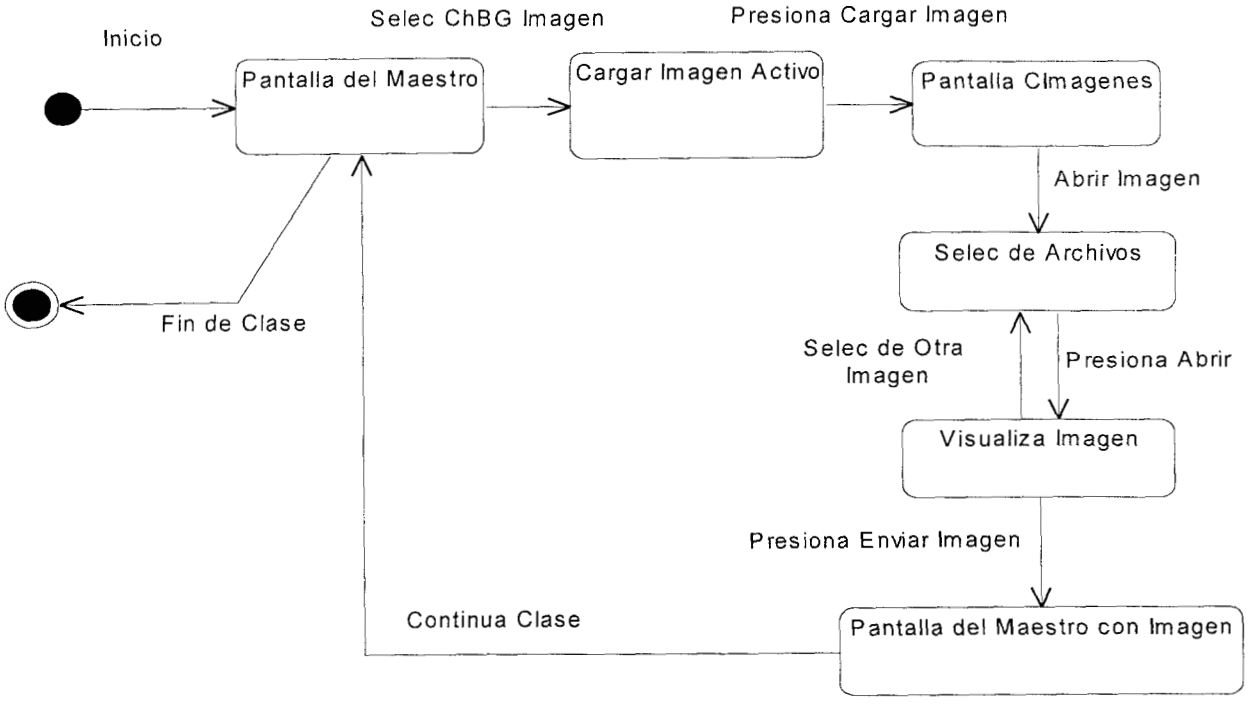


Diagrama de Colaboracion

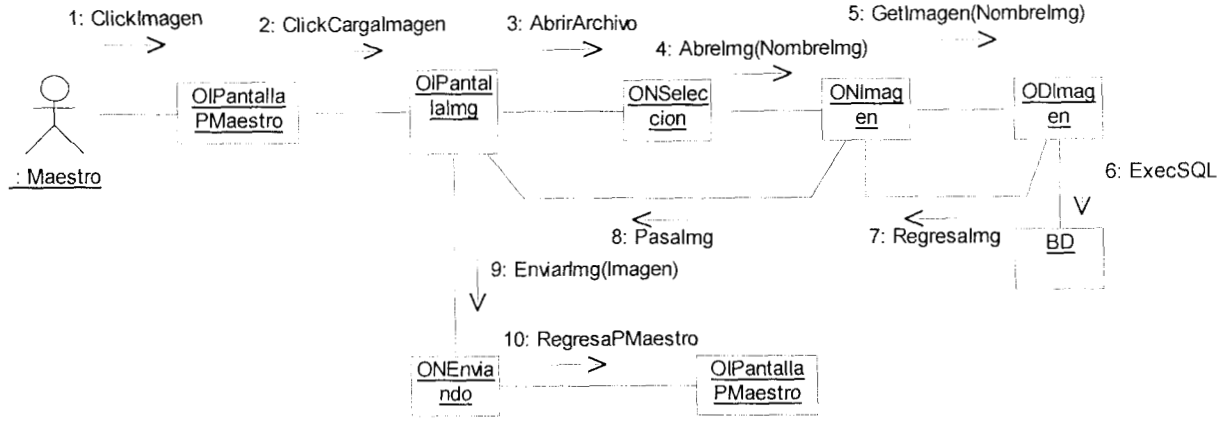
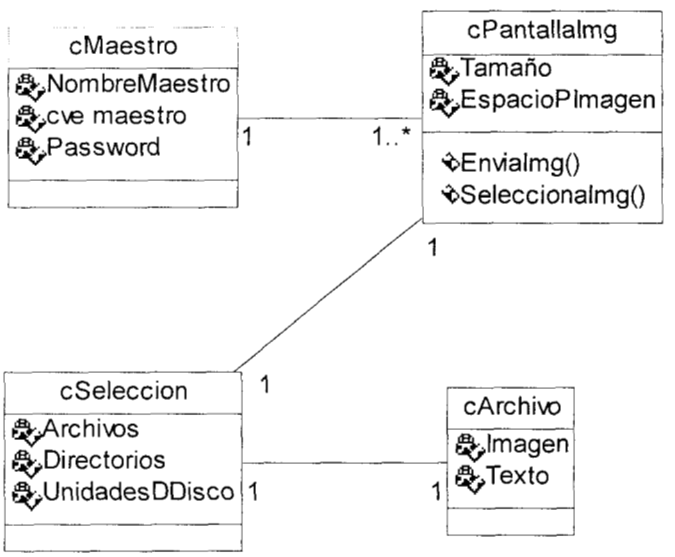


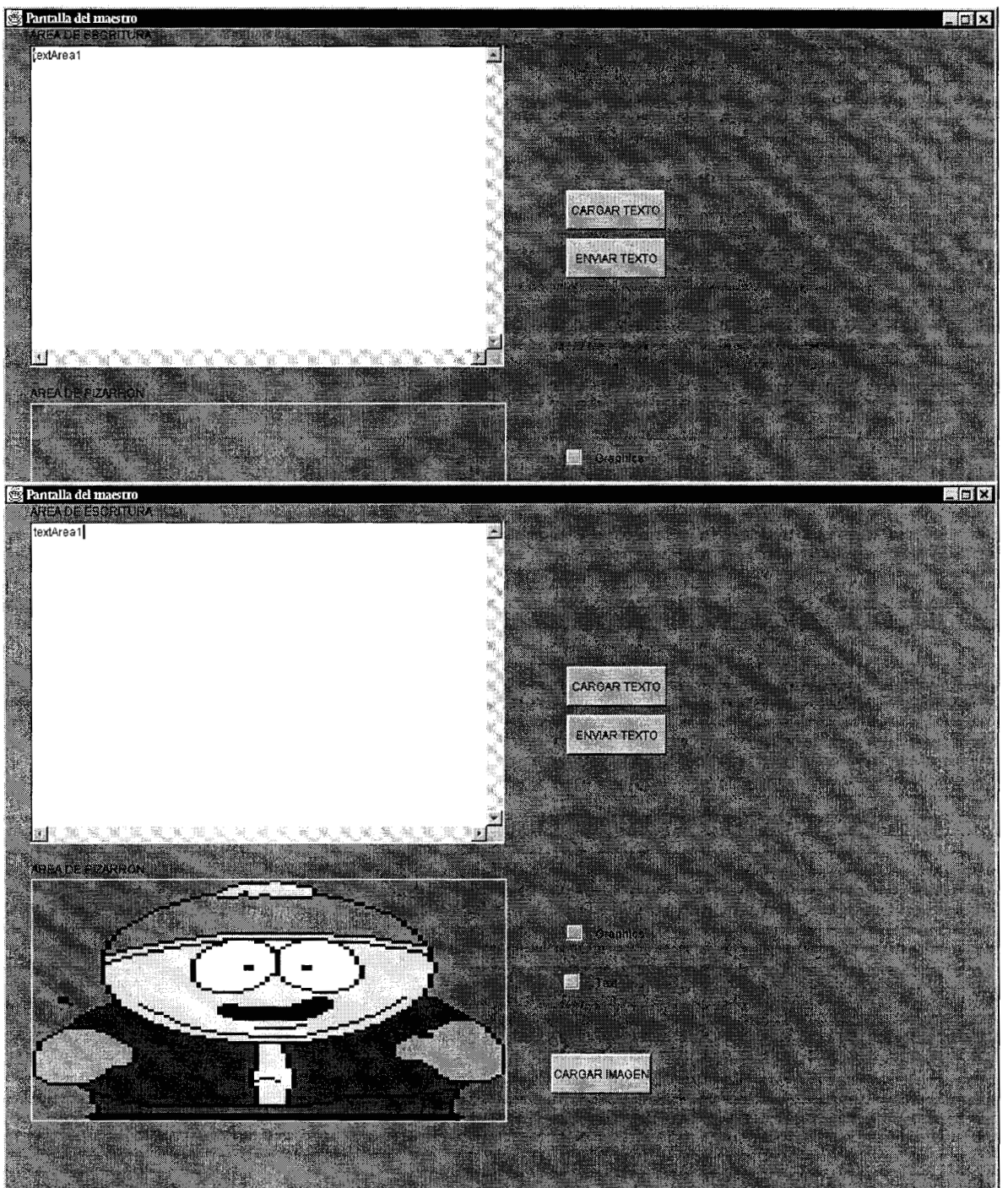
Diagrama de Secuencia

Modelo del Dominio del problema

Diagrama de Clases (Use Case CargaEnvialmagen)

Layouts de pantallas (Use Case CargaEnvialmagen):





Pantalla Climagenes

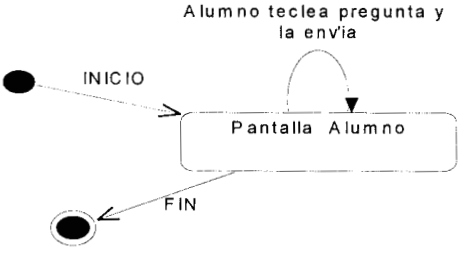
16. Use Case Alumno Pregunta

PASOS

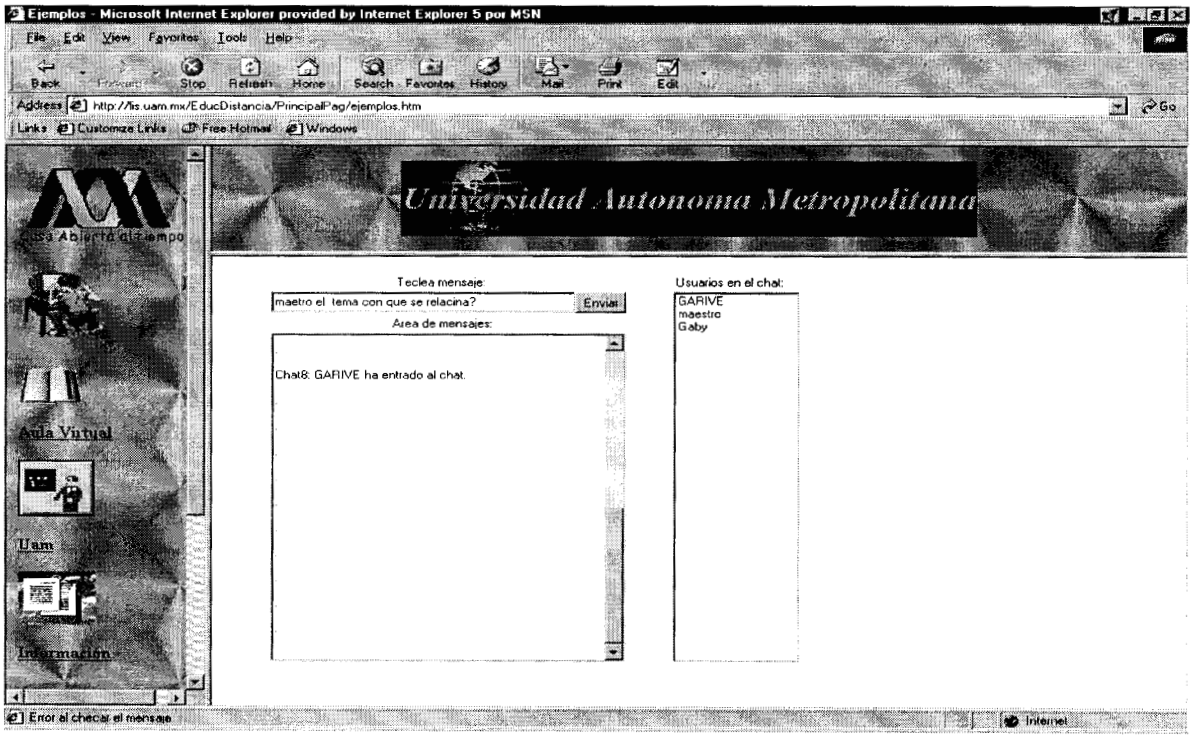
1.-Alumno teclea pregunta y presiona enviar.
2.-Sistema lleva pregunta a maestro.

Modelo de Interfaz

Diagrama de Estado(Use Case Alumno Pregunta)



Layouts de pantallas(Use Case Alumno Pregunta)

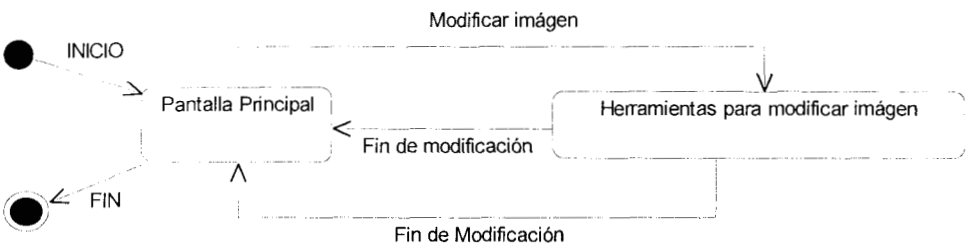


17. Use Case ModificarImágen

PASOS
1.-Maestro presiona modificar imágen.
2.-Sistema trae componentes de dibujo.
3.-Maestro modifica imágen.
4.-Maestro presiona salir de los componentes de la paleta(dejar de dibujar).

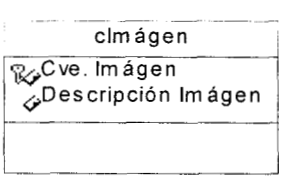
Modelo de Interfaz

Diagrama de Estado(Use Case ModificarImágen)



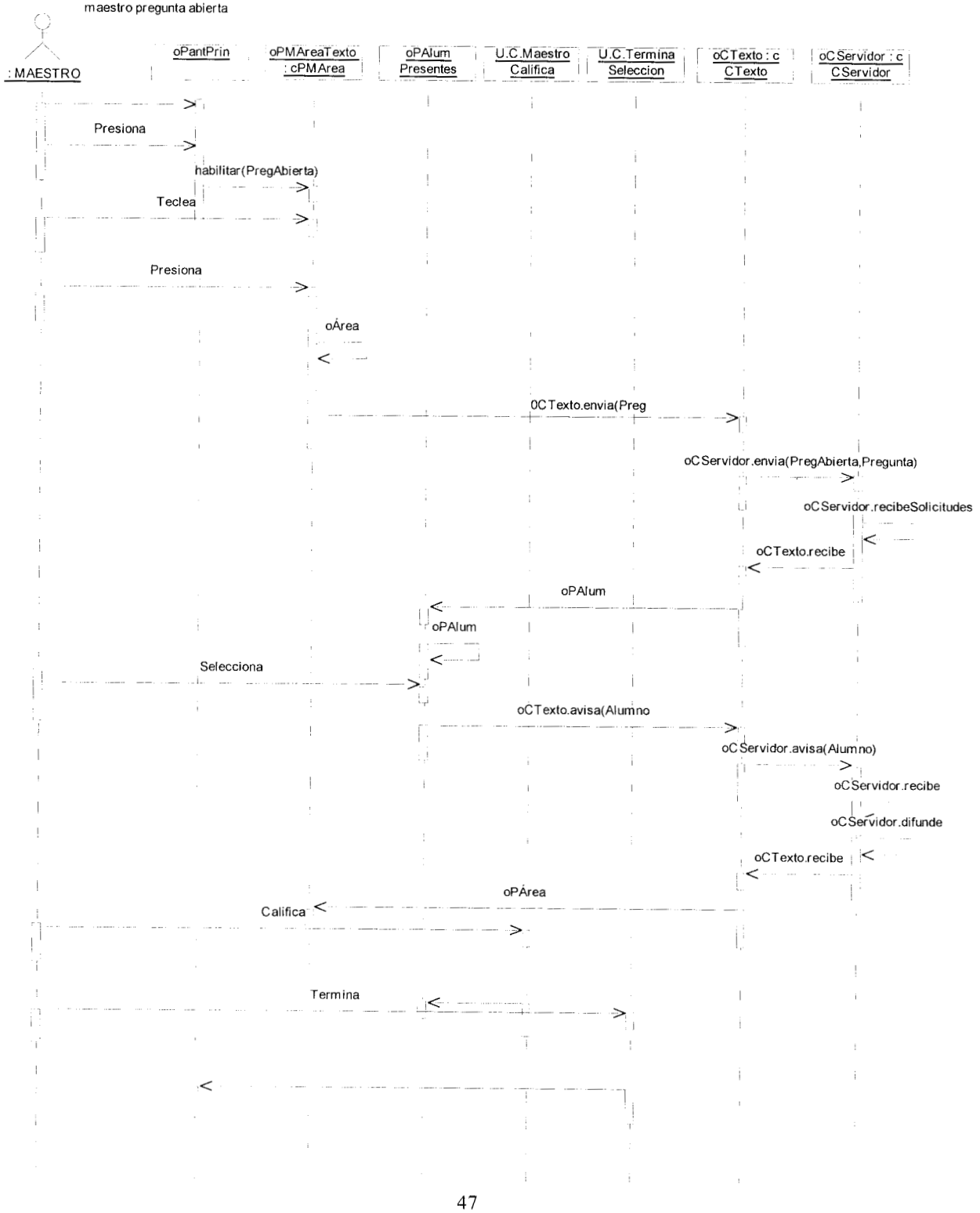
Modelo del Dominio del problema

Diagrama de Clases (Use Case ModificarImágen)

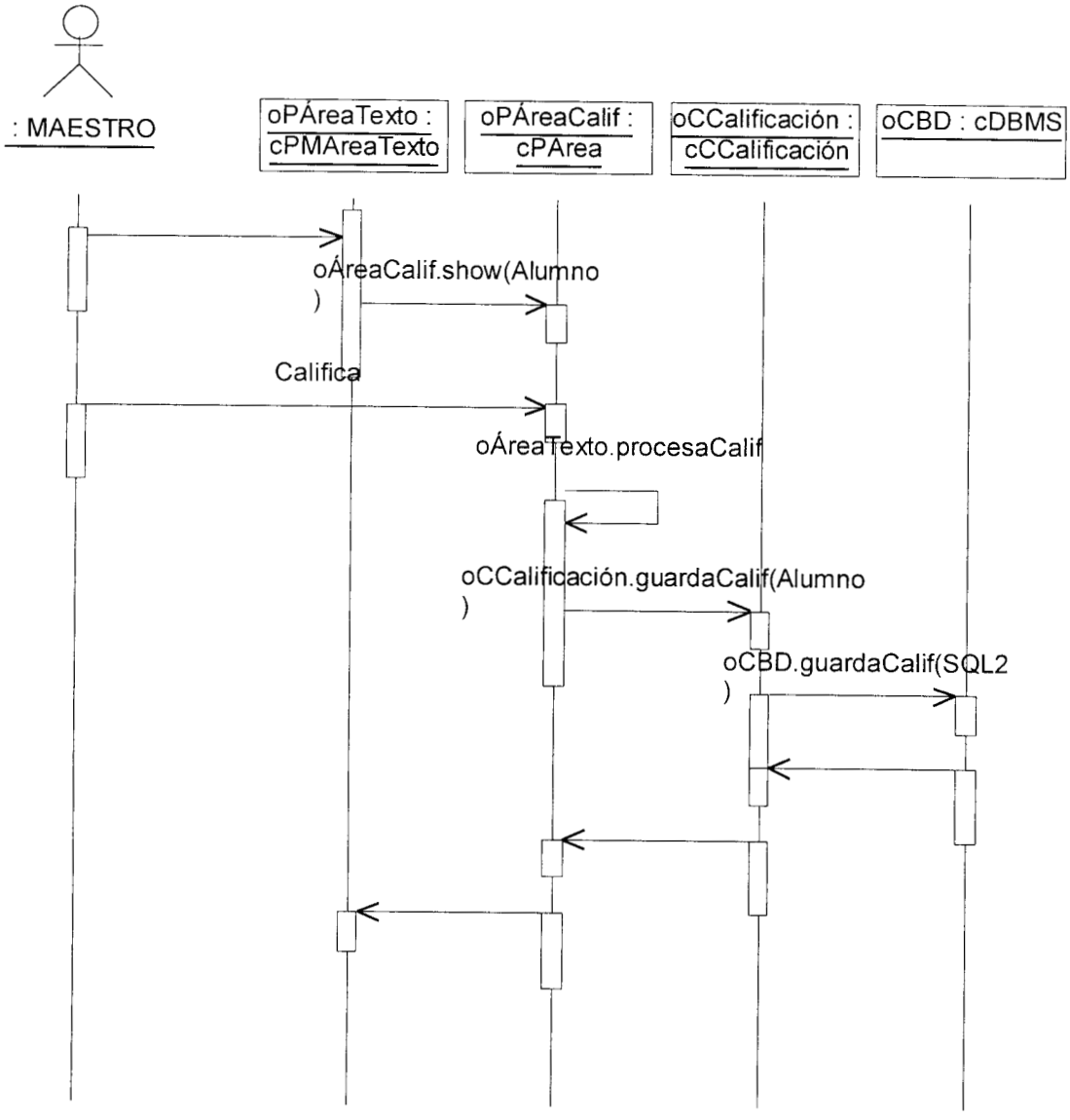


ANEXO3: Modelo de Diseño (Diagramas de Secuencia)

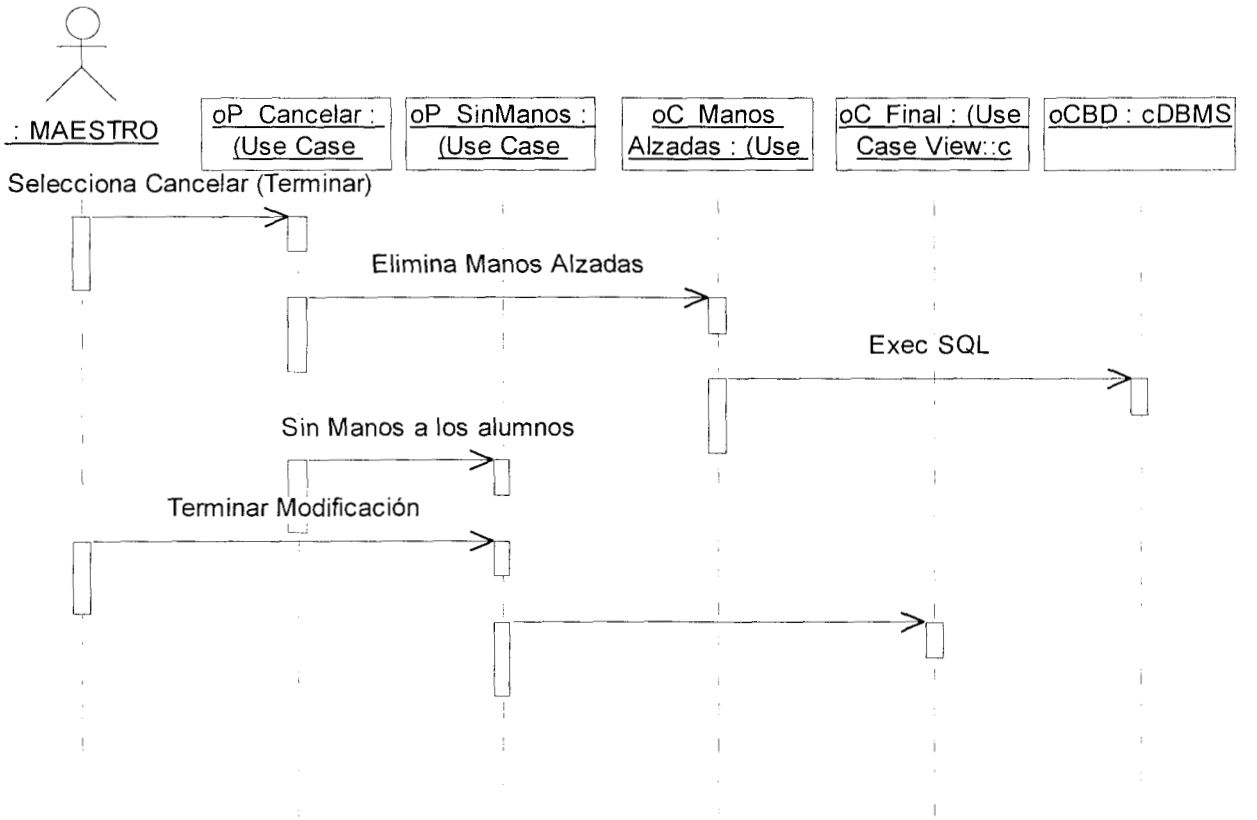
1. (Use Case Maestro Pregunta Abierta)



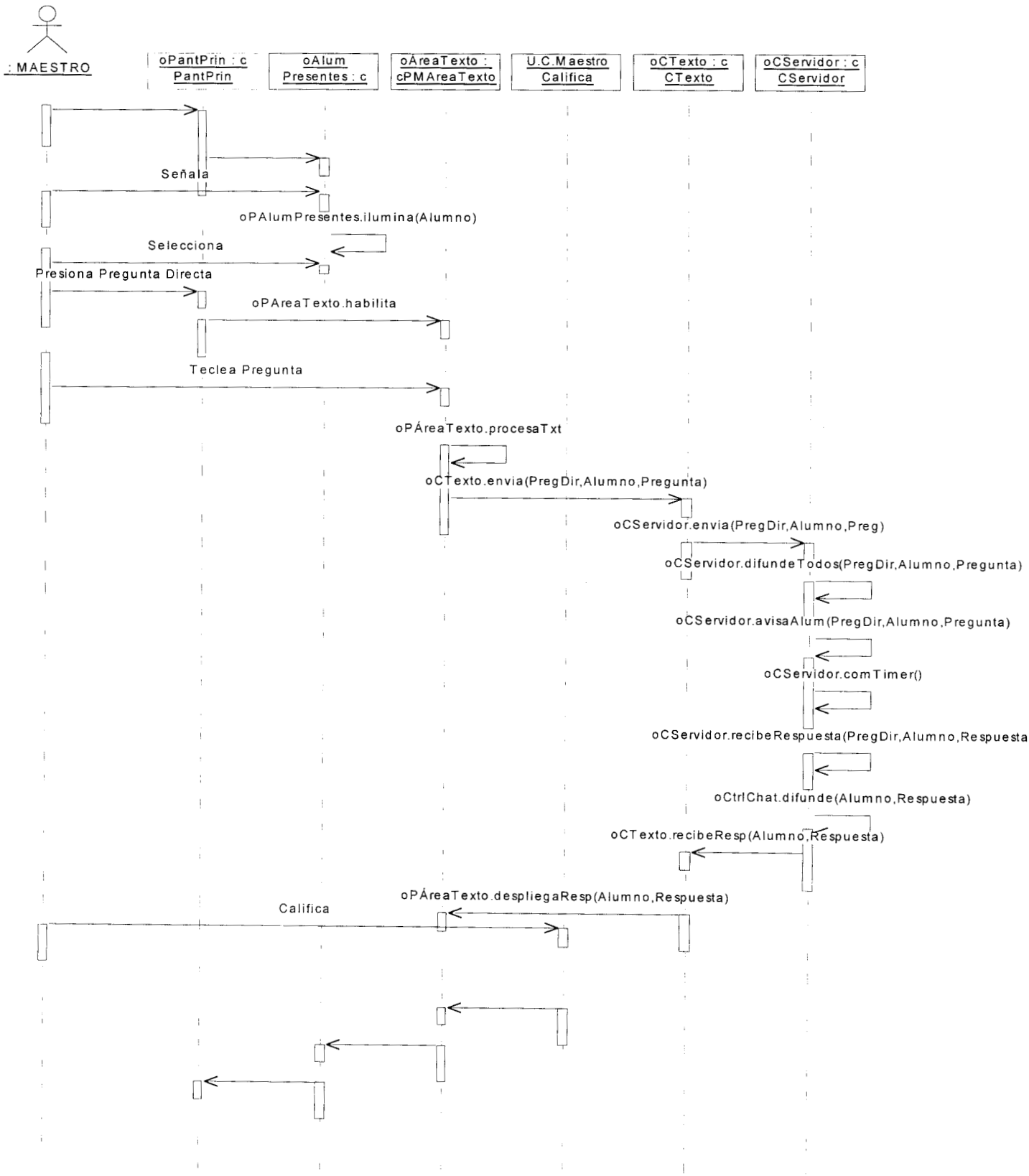
2. (Use Case Maestro Califica Respuesta)



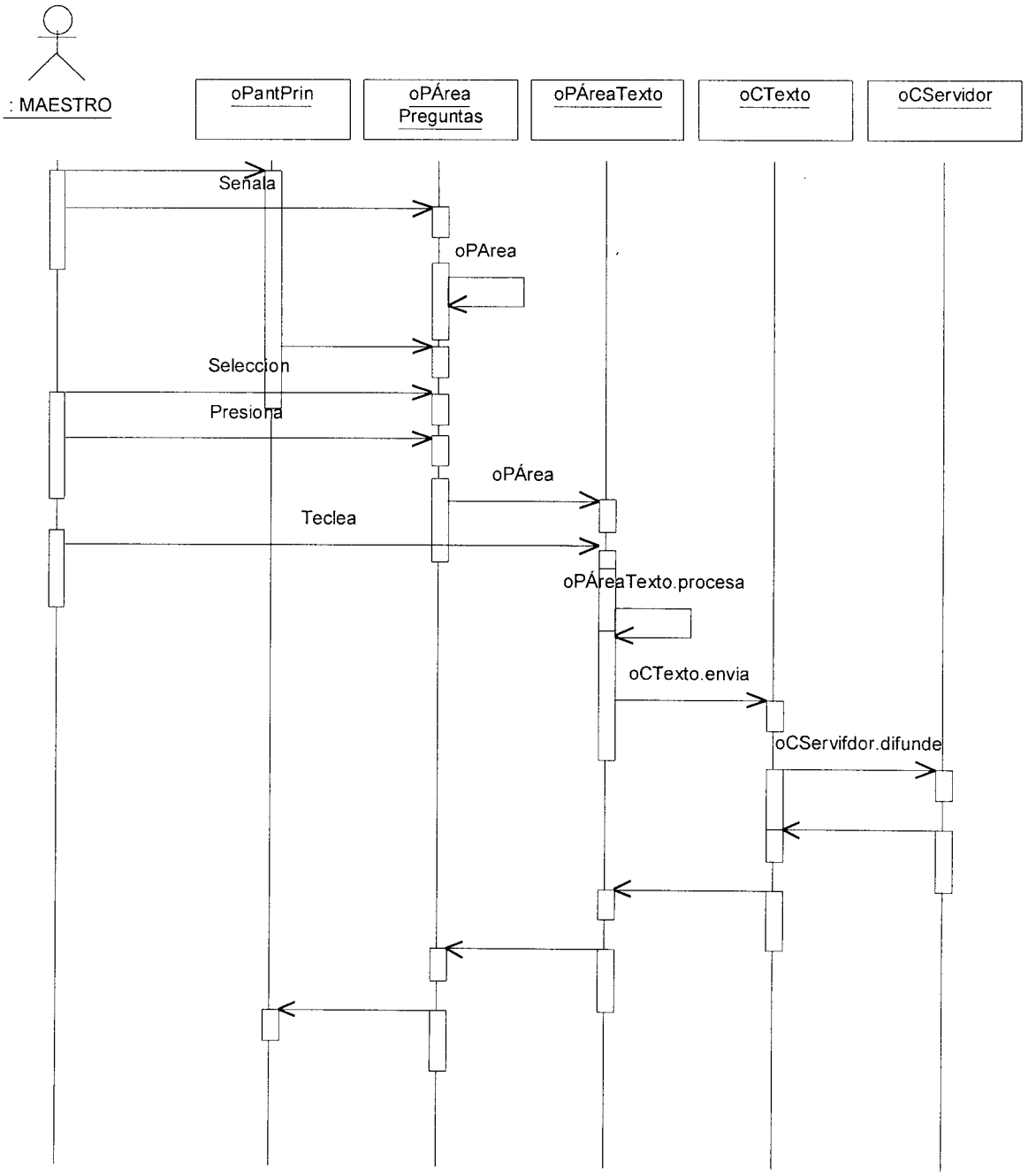
2. (Use Case Termina Selección Prgunta Abierta)



(Use Case Maestro Pregunta Directa)

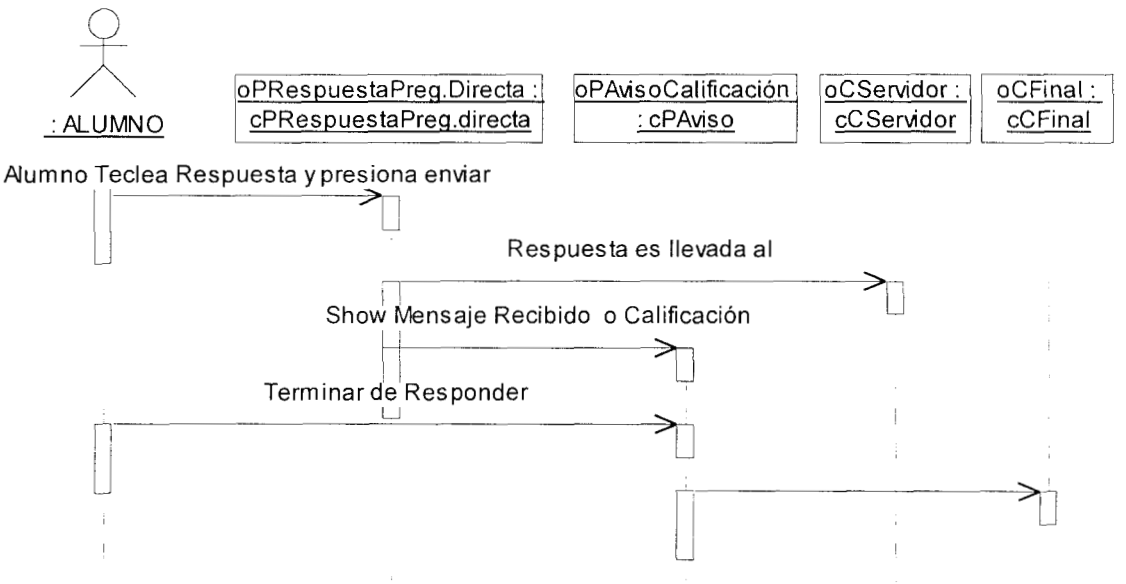


3. (Use Case Maestro Contesta Directamente)



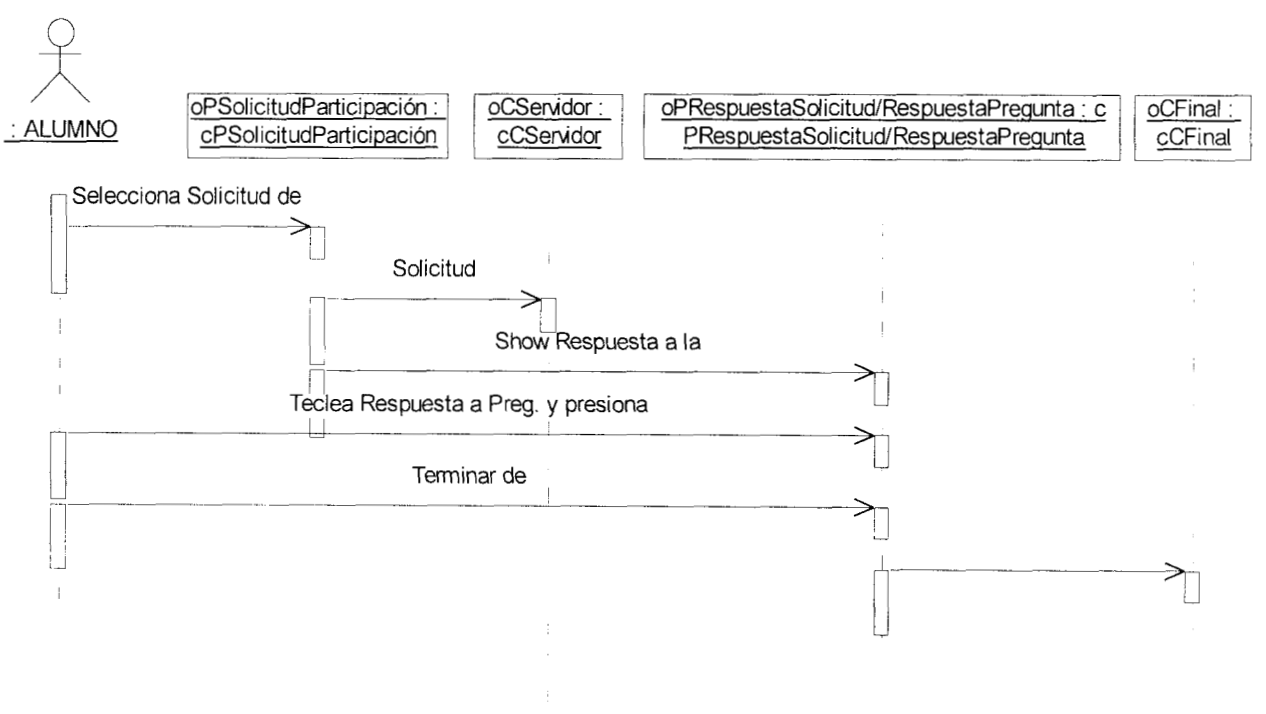
5. (Use Case ContestaConPredeterminada)

4. (Use Case AlumnoRespondePreguntaDirecta)



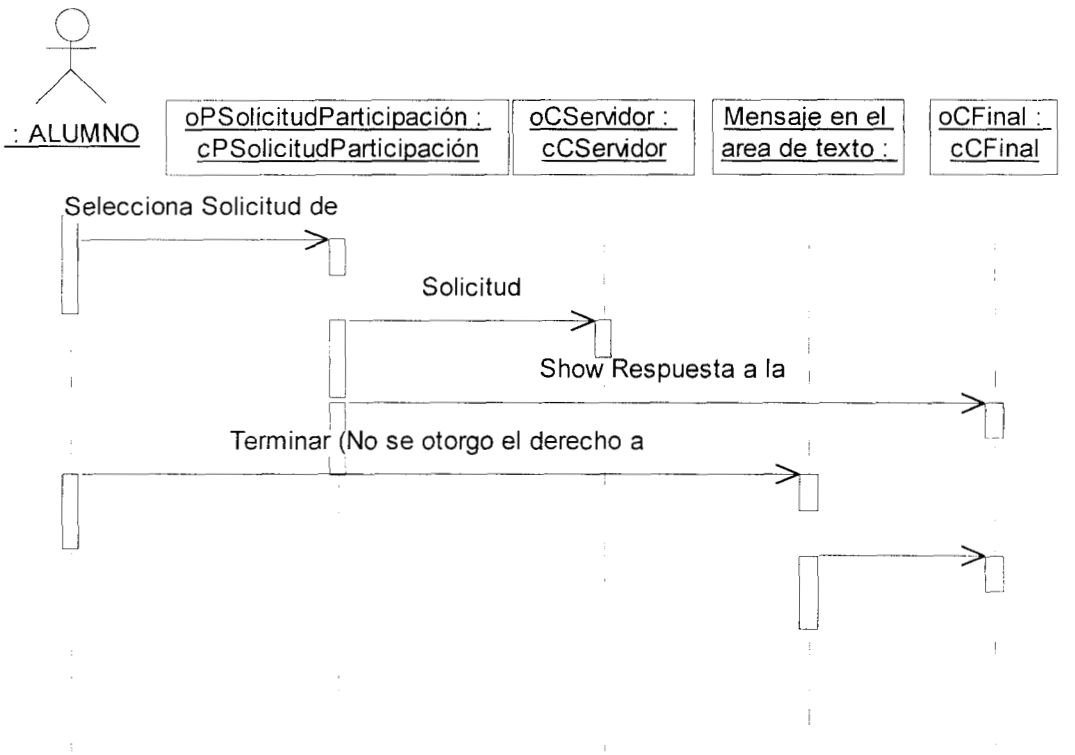
5. (Use Case AlumnoRespondePreguntaAbierta)

I

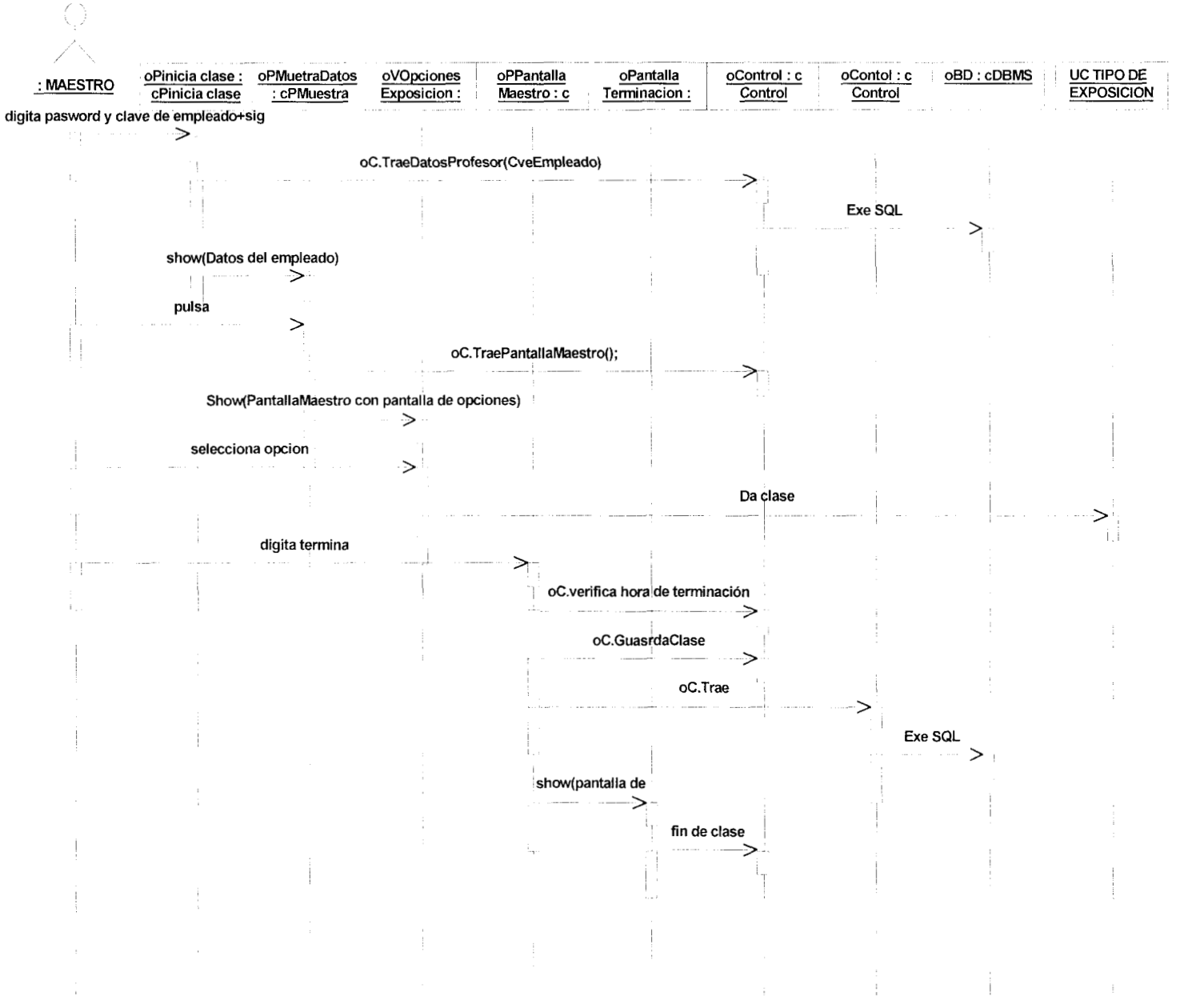


6. (Use Case AlumnoRespondePreguntaAbierta)

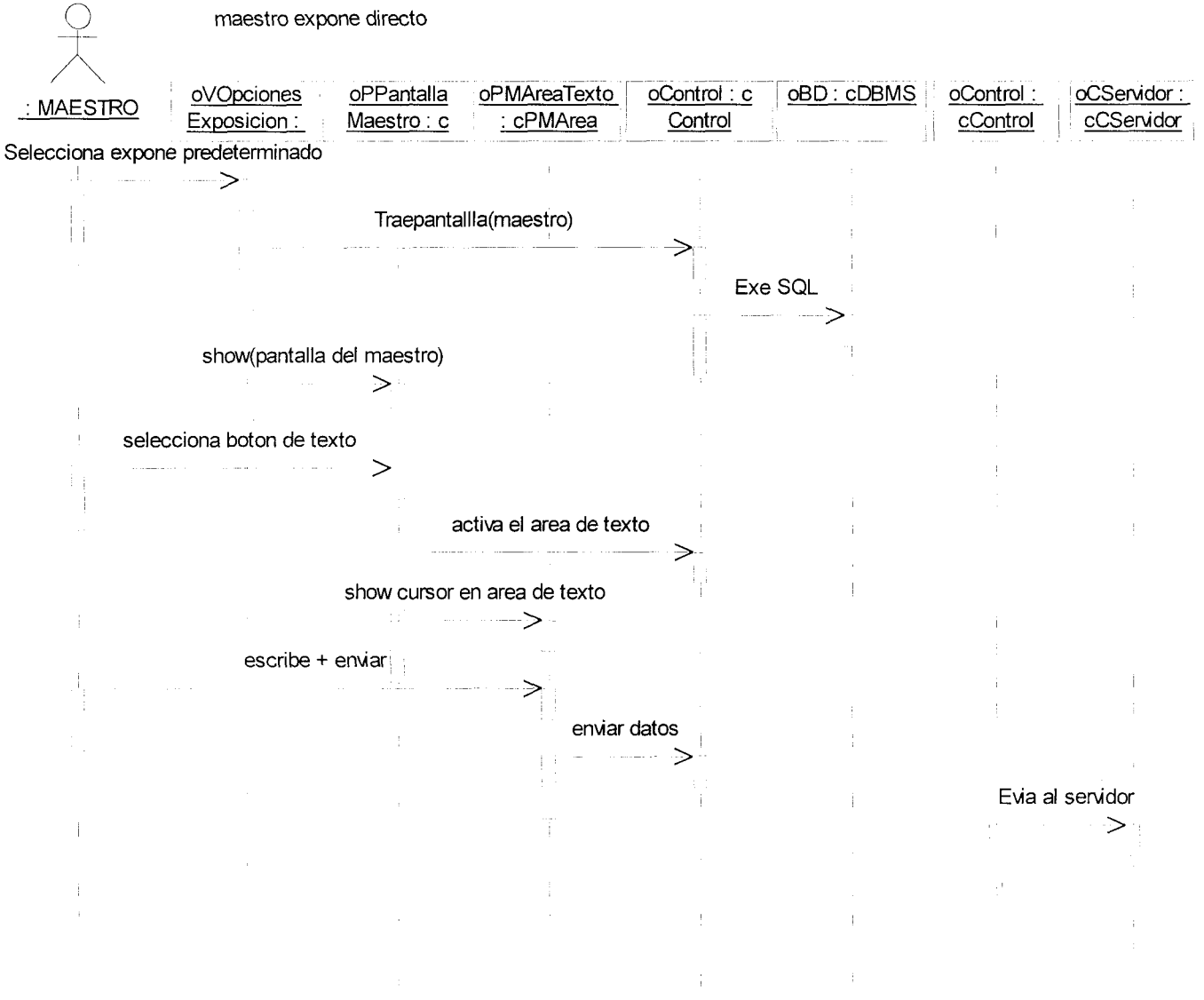
II



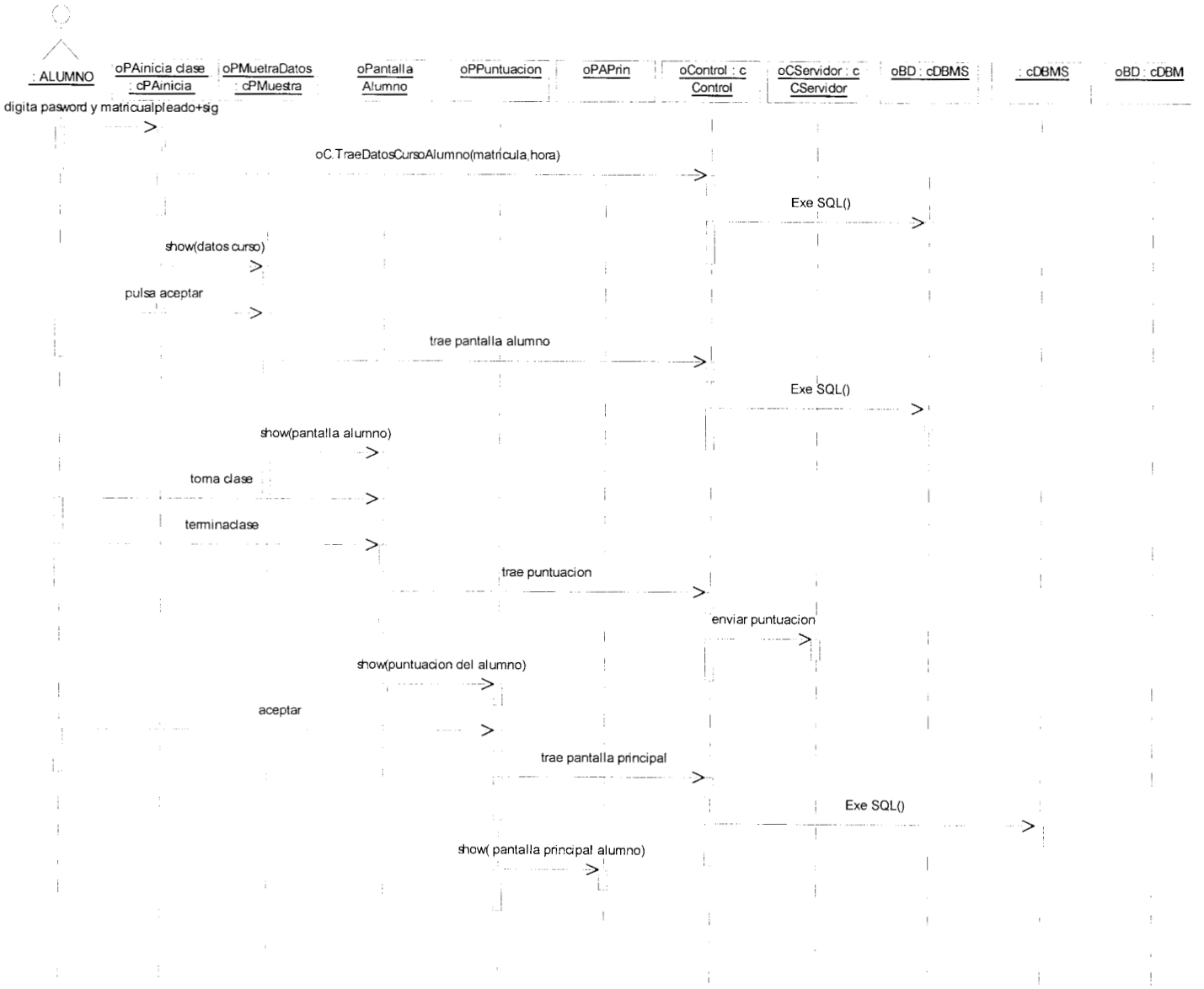
7. (Use Case Maestro entra a dar clase)



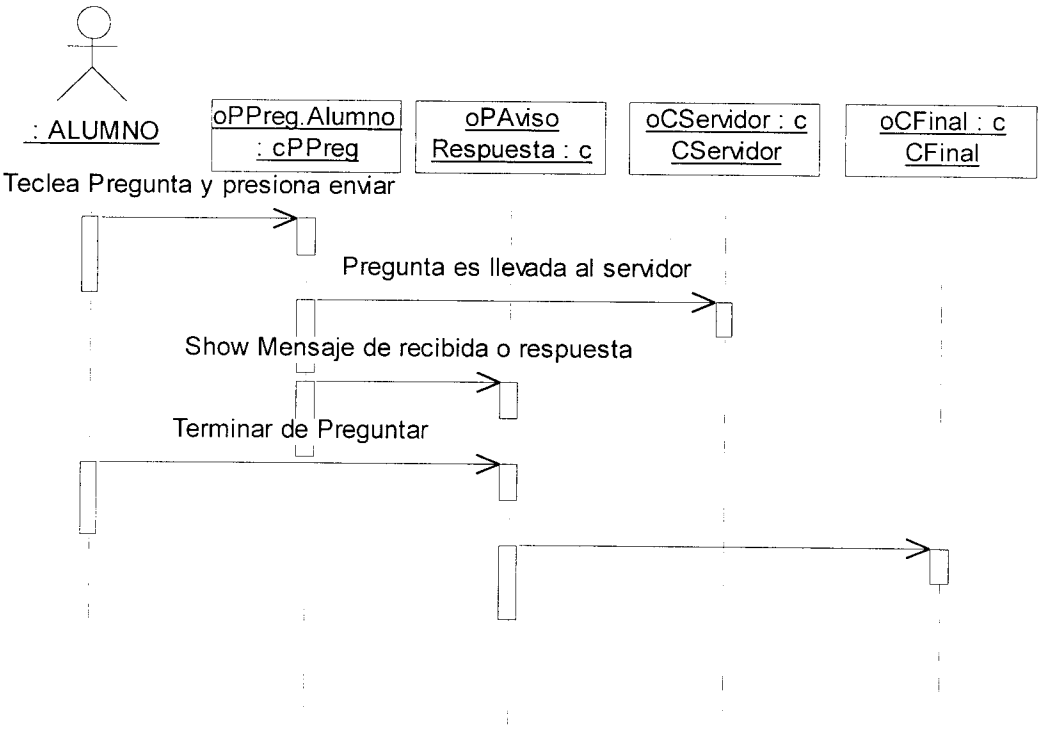
8. (Use Case Maestro Expone Directo)



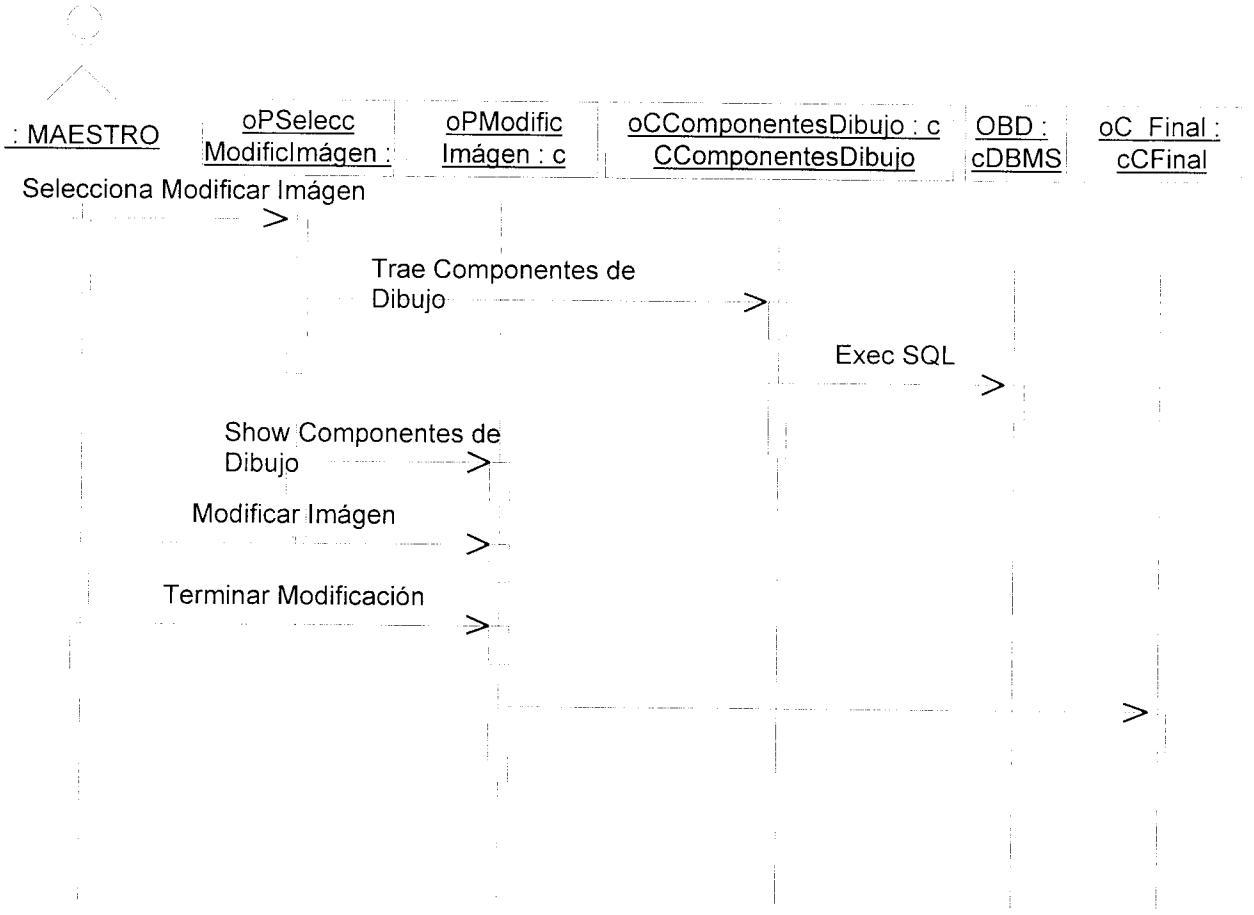
9. (Use Case Alumno Entra a clase)



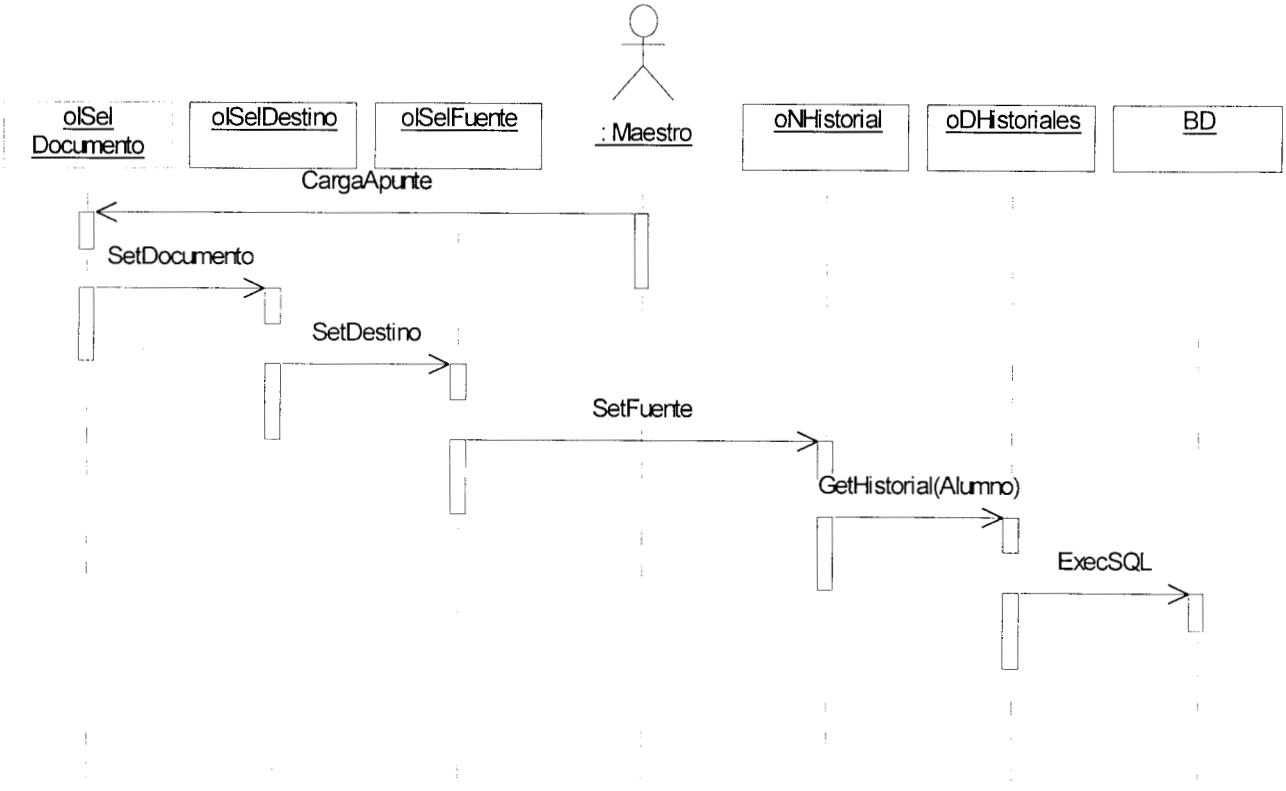
10. (Use Case Alumno Pregunta)



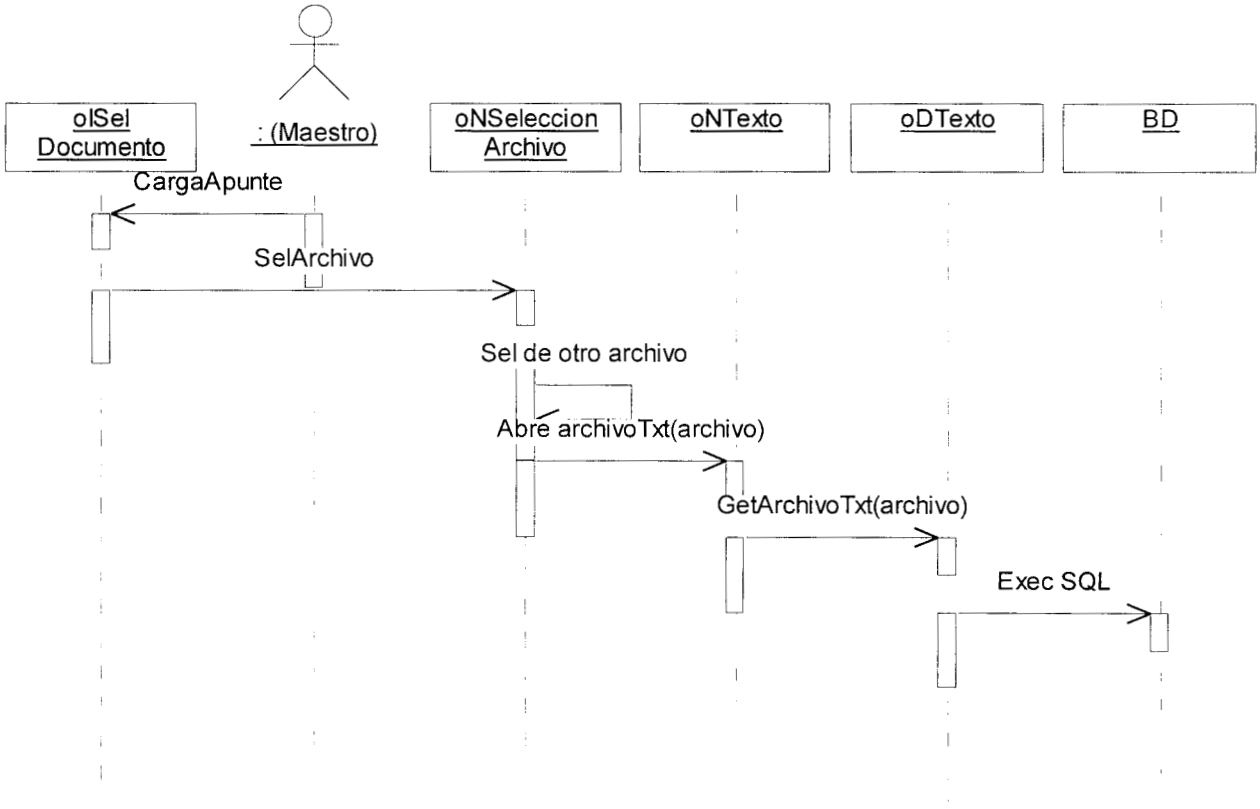
11. (Use Case Modifica Imagen)



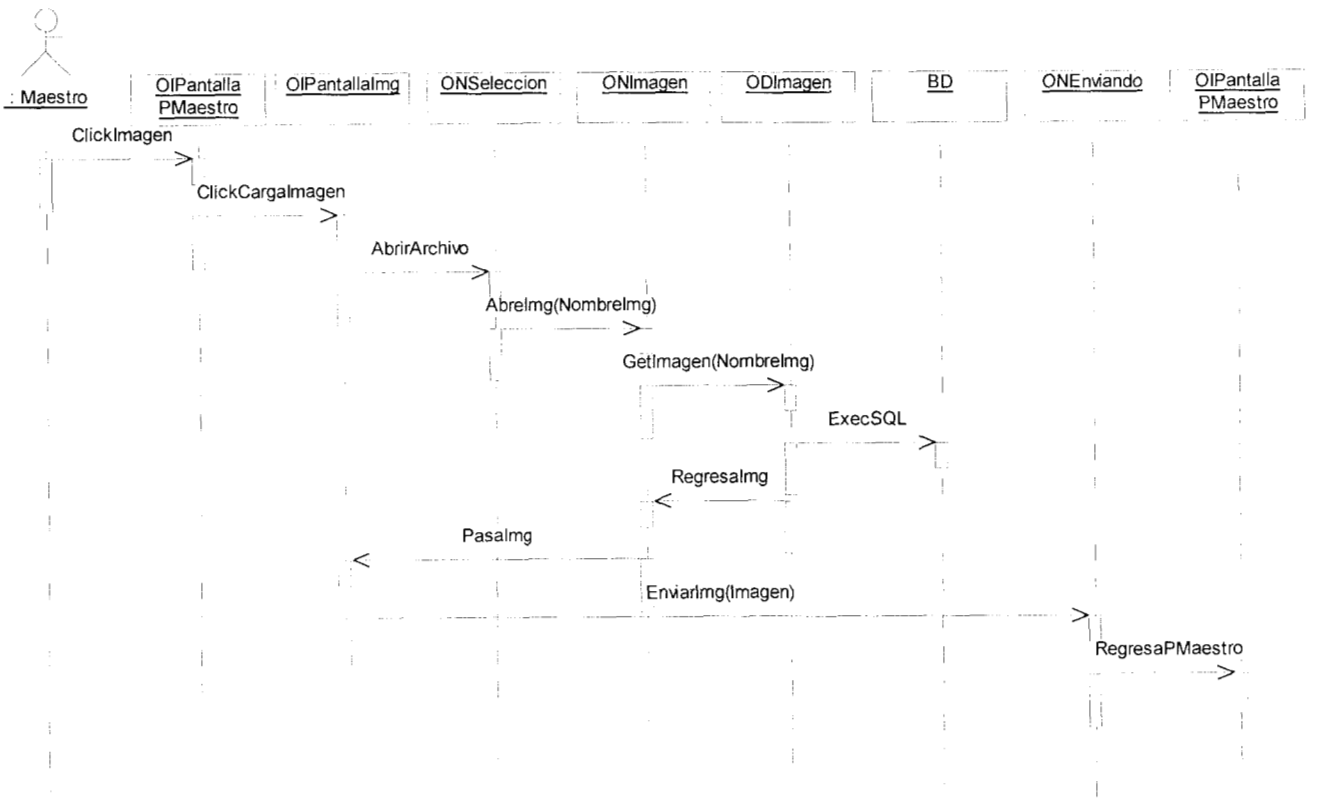
15. (Use Case Carga Envia como Documento)



16. **(Use Case Carga Envia Directamente)**



17. (Use Case Carga Envia Imagen)



4.2.4 Modelo de Construcción.

El modelo de construcción es en gran parte el diagrama de secuencia, pero con las relaciones que tiene con respecto a la codificación, este modelo de construcción representa en gran parte la codificación que se realizo en la implantación del sistema.

4.2.4.1 Introducción.

para obtener el modelo de construcción se tiene que tomar el modelo de análisis dinámico para generar un modelo de diseño que contemple aspectos de implantación y aspectos de implantación y aspectos de calidad, posteriormente se toma el modelo de diseño como entrada para la obtención del modelo de construcción.

Modelo de Construcción

A continuación se presenta la codificación que nos da en gran parte el modelo de construcción esta ,codificación es la original que nos presenta los Servlets, Applets, paginas y codificación mas importante con respecto al proyecto de Educación a Distancia.

```
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;
import borland.jbcl.layout.*;
import java.net.*;
import java.io.*;
import java.util.*;
import borland.jbcl.control.*;
import borland.jbcl.control.ImageControl;

public class PMaestro extends JFrame implements Runnable{

    private static final String SERVLET_PATH="/servlet/chat.ChatServlet";
    private static final String servletURL="http://lis-server01:8080";
    volatile private boolean bEntro;
    //Pendiente de capturar.
    String cveMaestro="1";
    String cveUEA="1";
    URL chatURL;
    URLConnection conexion;
    String usuario="maestro";
    boolean bRespondePreg=false;
    Thread HiloColector;
    //Construct the frame
    JMenuBar menuBar1 = new JMenuBar();
    JMenu menuFile = new JMenu();
    JMenuItem menuFileExit = new JMenuItem();
    JMenu menuHelp = new JMenu();
    JMenuItem menuHelpAbout = new JMenuItem();
    JLabel statusBar = new JLabel();
    XYLayout xYLayout1 = new XYLayout();
    TextArea txtaAreaTexto = new TextArea();
```

```
        List listaUsuarios = new List();
        Button btnEnviar = new Button();
        Button btnCargar = new Button();
        List listaPreguntas = new List();
        CheckboxGroup chkboxgImgTxt = new CheckboxGroup();
        Checkbox chkboxImg=new Checkbox("Imágen",chkboxgImgTxt,false);
        Checkbox chkboxTxt=new Checkbox("Texto",chkboxgImgTxt,true);
        Button btnResponder = new Button();
        Label lblAreatexto = new Label();
        Label lblListaAlumnos = new Label();
        Label lblListaPreguntas = new Label();
    TextArea Area2= new TextArea();
    TextControl textControl1 = new TextControl();
    StringInput stringInput1 = new StringInput();
    Button BGuardar = new Button();
    ButtonControl btnCargarHis = new ButtonControl();
    List Lista2 = new List();
    Panel panel1 = new Panel();
    ImageControl imageControl1 = new ImageControl();
    ImageControl imageControl3 = new ImageControl();
    ImageControl imageControl2 = new ImageControl();
    ImageControl imageControl4 = new ImageControl();


        /*inicia pantalla del maestro*/
        public PMaestro() {
                enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try  {

                        iniciaHilo();
                        jbInit();
                        entrar();// Funcion de entrada en el servidor
                }
                catch (Exception e) {
                        e.printStackTrace();
                }
        }
/*Este metodo controla los procesos (hilos)  dado un cierto tiempo*/
        public void run(){
                while(true){
                                try{
                                        //Funciones leidas del servlet .
                                        colectaPreguntas();
                                        colectaLista();
                                        HiloColector.sleep(100);
                                }
                                catch(InterruptedException e){
                                        e.printStackTrace();
                                }
                }

        }
```

```
                Area2.setBackground(new Color(178, 255, 255));
                textControl1.setText("   Texto Eviado");
                BGuardar.setBackground(new Color(201, 255, 232));
                BGuardar.setLabel("Guardar");
                btnCargarHis.setBackground(new Color(201, 255, 232));
                btnCargarHis.setLabel("Historial");
                imageControl1.setImageName("\\\\Lis-
server01\\csh\\EducDistancia\\Profesor.gif");
                imageControl3.setImageName("\\\\Lis-
server01\\csh\\EducDistancia\\libro2.gif");
                imageControl4.setImageName("\\\\Lis-
server01\\csh\\EducDistancia\\BajaInf.gif");
                btnCargarHis.addActionListener(new java.awt.event.ActionListener()
                {
                        public void actionPerformed(ActionEvent e)
                        {
                                btnCargarHis_actionPerformed(e,Lista2);
                        }
                });
                BGuardar.addMouseListener(new java.awt.event.MouseAdapter()
                {
                        public void mouseClicked(MouseEvent e)
                        {
                                BGuardar_mouseClicked(e);
                        }
                });
                xYLayout1.setWidth(1036);
                menuFile.setText("File");
                menuFileExit.setText("Exit");
                menuFileExit.addActionListener(new ActionListener()  {
                        public void actionPerformed(ActionEvent e) {
                                fileExit_actionPerformed(e);
                        }
                });
                menuHelp.setText("Help");
                menuHelpAbout.setText("About");
                menuHelpAbout.addActionListener(new ActionListener()  {
                        public void actionPerformed(ActionEvent e) {
                                helpAbout_actionPerformed(e);
                        }
                });
                menuFile.add(menuFileExit);
                menuHelp.add(menuHelpAbout);
                menuBar1.add(menuFile);
                menuBar1.add(menuHelp);
                this.setJMenuBar(menuBar1);
                this.getContentPane().add(statusBar, new XYConstraints(307, 650, 751, -
1));
                this.getContentPane().add(txtaAreaTexto, new XYConstraints(20, 220, 498,
166));
                this.getContentPane().add(listaUsuarios, new XYConstraints(642, 32, 235,
328));
```

```
                this.getContentPane().add(btnEnviar, new XYConstraints(37, 612, 123,
30));
                this.getContentPane().add(btnCargar, new XYConstraints(198, 612, 77,
30));
                this.getContentPane().add(chkboxImg, new XYConstraints(43, 572, 113,
20));
                this.getContentPane().add(chkboxTxt, new XYConstraints(43, 548, 113,
20));
                this.getContentPane().add(listaPreguntas, new XYConstraints(645, 425,
356, 205));
                this.getContentPane().add(btnResponder, new XYConstraints(785, 646,
105, 37));
                this.getContentPane().add(lblAreatexto, new XYConstraints(162, 196, 139,
17));
                this.getContentPane().add(lblListaAlumnos, new XYConstraints(676, 9, 181,
14));
                this.getContentPane().add(lblListaPreguntas, new XYConstraints(780, 405,
126, 18));
                this.getContentPane().add(Area2, new XYConstraints(22, 34, 502, 158));
                this.getContentPane().add(textControl1, new XYConstraints(189, 9, 129,
24));
                this.getContentPane().add(BGuardar, new XYConstraints(533, 35, 72, 28));
                this.getContentPane().add(btnCargarHis, new XYConstraints(900, 35, 105,
22));
                this.getContentPane().add(panel1, new XYConstraints(743, 53, 42, 37));
                this.getContentPane().add(imageControl1, new XYConstraints(919, 86, 64,
85));
                this.getContentPane().add(imageControl3, new XYConstraints(101, 648,
31, 47));
                this.getContentPane().add(imageControl4, new XYConstraints(214, 648,
37, 47));

        }
 /*fin de jbinit*/

 /*inicio de funciones  para jbinit*/
        protected void iniciaHilo(){
                int prioridadActual=Thread.currentThread().getPriority();
                int
prioridadNueva=prioridadActual==Thread.MIN_PRIORITY?Thread.MIN_PRIORITY:priorid
adActual-1;
                HiloColector=new Thread(this,"AplicacionColectora");
                HiloColector.setDaemon(true);
                HiloColector.setPriority(prioridadNueva);
                HiloColector.start();
                p("Iniciando colección...");

        }
//File | Exit action performed

        public void fileExit_actionPerformed(ActionEvent e) {
                if(HiloColector!=null&&HiloColector.isAlive()){
```

```
                    HiloColector.stop();

      ///cuando cierra el chat se destruye el hilo colector

                }
                abandonar();
  HiloColector=null;
  System.exit(0);
  HiloColector.destroy();
      }


      //Conexion al chat especifico

      protected  void  entrar() throws MalformedURLException{
            chatURL=new URL(servletURL);
            usuario="maestro";
            String
queryString=SERVLET_PATH+"?modo=nuevo_usuario&usuario="+URLEncoder.encode(
usuario);
            p("Tratando de entrar.");
    try{
      conexion=(new URL(chatURL,queryString)).openConnection();
      conexion.setDefaultUseCaches(false);
      conexion.setUseCaches(false);
      conexion.setDoInput(true); //Flujo de Entrada
      conexion.setDoOutput(false);
      conexion.connect();
      p("Realizando conexion con"+conexion);
      BufferedReader          in          =new          BufferedReader(new
InputStreamReader(conexion.getInputStream()));
      String respuesta=in.readLine();
                  if(respuesta.startsWith("+")){  //Detecta si se pudo conectar
        setEntro(true);
        p("Entro como:"+usuario);
        repaint();
      }
      else{
        p("Error al intentar conectarse: "+respuesta);
        System.exit(0);
      }
    }
    catch(MalformedURLException e){
      p("Error en URL:");
      System.err.println(e);
      e.printStackTrace(System.err);
    }
            catch(IOException e2){
                  p("Error IO al intentar conexion");
                  System.err.println(e2);
                  e2.printStackTrace(System.err);
            }
```

```
        }

        protected void setEntro(boolean set){
                bEntro=set;
        }
        protected boolean yaEntro(){
                return bEntro;
        }
 /*metodo que elimina a un alumno de la clase*/
 protected void abandonar() {
          if (!yaEntro() || usuario == null) return;
          String queryString = SERVLET_PATH +
"?modo=borra_usuario&usuario="+URLEncoder.encode(usuario);
          try {
                        conexion = (new URL(chatURL,queryString)).openConnection();
                        conexion.setUseCaches(false);
                        conexion.setDoInput(true);
                        conexion.setDoOutput(false);
                        conexion.connect();
                        BufferedReader in = new BufferedReader(new
InputStreamReader(conexion.getInputStream()));
                        String respuesta = in.readLine();
                        if (respuesta.startsWith("+")) {
                                        setEntro(false);
                                        p("Usuario " + usuario + " ha salido del chat");
                }
       else {
                                        p("Error al salir" + respuesta);
                                        System.err.println("Error al salir" +respuesta);
                                }
                                in.close();
                } catch (MalformedURLException e2) {
                                System.err.println(e2);
                                e2.printStackTrace(System.err);
                                p("Error al intentar salir");
                } catch (IOException e1) {
                                System.err.println(e1);
                                e1.printStackTrace(System.err);
                                p("Error al salir");
                }
        }

/*metodo enviar, envia a todos los alumnos los mansajes que el maestro escribe
en el area de texto*/
          protected  void enviar() {
            String mensaje;
     String nulo;
                int inicio;
                int fin;
     //asignaciones
     mensaje= txtaAreaTexto.getSelectedText();
     inicio=txtaAreaTexto.getSelectionStart();
```

```
                fin=txtaAreaTexto.getSelectionEnd();

    if (mensaje.equals("")) {
                        mensaje = txtaAreaTexto.getText();
                        if (mensaje.equals("")) return;
    }
    p("Enviando mensaje.");
    //realiza la conexion para enviar mensajes
                try {
                                conexion=null;
                                conexion                        =                       (new
URL(chatURL,SERVLET_PATH)).openConnection();
                                conexion.setUseCaches(false); //Para este
                                conexion.setDefaultUseCaches(false);
                                conexion.setDoInput(true); //Manda objetos con el metodo
POST
                                conexion.setDoOutput(true);
                                conexion.setRequestProperty("Content-
Type","application/octet-stream");
                                OutputStream os=conexion.getOutputStream();
                                ObjectOutputStream                              out=new
ObjectOutputStream(/*conexion.getOutputStream()*/os);//Envia los objetos
                                out.writeObject(mensaje);
                                out.flush();
                                BufferedReader    in    =    new    BufferedReader(new
InputStreamReader(conexion.getInputStream()));
                                String response = in.readLine();
                                if (response.startsWith("+")) {//Acepto el mensaje

        if(inicio!=fin)    //si hay algo seleccionado en el area de texto para enviar
          txtaAreaTexto.replaceText("",inicio,fin) ;
        else
          txtaAreaTexto.setText(""); //AREA DE TEXTO1

        p("Mensaje enviado");
                        Area2.append("\n");
                        Area2.append(mensaje);

                                }
        else {
                                        p("Error al enviar mensaje" + response);

                                }
                                in.close();
                                out.close();
                                bRespondePreg=false;

                } catch (MalformedURLException e2) {
                                p("Error al enviar mensaje");
                                //System.err.println(e2);
                                e2.printStackTrace(System.err);
                } catch (IOException e1) {
```

```
                    p("Error al enviar mensaje IO");
                    //System.err.println(e1);
                    e1.printStackTrace(System.err);
                }
        }

/*metodo donde el maestro recibe informacion del servidor*/
        protected void  colectaPreguntas() {
    String queryString = SERVLET_PATH +
"?modo=colecta&usuario="+URLEncoder.encode(usuario);
                try {
                    BufferedReader in = new BufferedReader(new
InputStreamReader(new URL(chatURL,queryString).openStream()));
                    String nextLine = in.readLine();
                    if (!nextLine.startsWith("+")) { //Si acepto la peticion de
preguntas
                        p("Error obteniendo mensajes del servidor");
                        return;
                    }
                    nextLine = in.readLine();
                    if (nextLine!=null){
                        p("usuarios colectados");
                        while (nextLine != null && !nextLine.equals(".")) {
//Recibe Preguntas
                            System.err.println(nextLine);
                            listaPreguntas.addItem(nextLine);
                            repaint();
                            nextLine = in.readLine();
                        }
                    }
                    else p("ningun usuario");
                    in.close();
                } catch (IOException e) {
                    System.err.println(e);
                    e.printStackTrace(System.err);
                    p("Error al checar el mensaje");
                }

        }

 //Pregunta por la lista de usuarios
        protected void  colectaLista() {
            String queryString = SERVLET_PATH + "?modo=lista";
            Vector usuarios = new Vector();
            try {
                System.out.println("chatURL: " + chatURL);
                System.out.println("queryString: " + queryString);
                URL listaURL = new URL(chatURL,queryString);
                System.out.println("listaURL: " + listaURL);
                URLConnection listaCon = listaURL.openConnection();
                listaCon.setDefaultUseCaches(false);
                listaCon.setUseCaches(false);
```

```
                    listaCon.connect();
                    BufferedReader in = new BufferedReader(new
InputStreamReader(listaCon.getInputStream()));
                    String sigLinea = in.readLine();
                    if (!sigLinea.startsWith("+")) { //acepto la peticion de lista
                            p("Error obteniendo lista de usuarios del
servidor"+sigLinea);
                            return;
                    }
                    sigLinea = in.readLine();
                    while (sigLinea != null && !sigLinea.equals(".")) { //Recolecta lista
                            p("Leyendo usuario: "+sigLinea);
                            usuarios.addElement(sigLinea);
                            sigLinea = in.readLine();
                    }
                    if (!usuarios.isEmpty()) { //Hay usuarios
                            listaUsuarios.removeAll();
                            int size = usuarios.size();
                            for (int i = 0; i < size; i++) {
                                    listaUsuarios.addItem((String)usuarios.elementAt(i));
                            }
                    }
                    else {
                            listaUsuarios.removeAll();
                            listaUsuarios.addItem("Ningun usuario dentro");
                    }
                    repaint();
                    in.close();

            } catch (IOException e) {
                            System.err.println(e);
                            e.printStackTrace(System.err);
                            p("Error al obtener lista"+e);
            }

        }

        ////////////////
//Overriden so we can exit on System Close

        protected void processWindowEvent(WindowEvent e) {
                super.processWindowEvent(e);
                if (e.getID() == WindowEvent.WINDOW_CLOSING) {
                        fileExit_actionPerformed(null);
                }
        }
 /*manda mensajes a la consola*/
        protected void p(String mensaje){
                System.out.println(mensaje);
        }
```

```
        protected void abreCargaTxt(){
                        Frame f=new Frame();
                        String linea;
                        String archivo;
                        FileDialog fd = new FileDialog(f,"ABRIR ARCHIVO");
                        fd.show();
                        archivo= fd.getDirectory()+fd.getFile();
                        try{
                                FileInputStream is = new FileInputStream(archivo);
                                BufferedReader ds =new BufferedReader(new
InputStreamReader(is));
                                while ((linea=ds.readLine())!= null)
                                {
                                        txtaAreaTexto.append(linea);
                                        txtaAreaTexto.append("\n");
                                }
                               ds.close();
                        }
                        catch(IOException e){
                          System.out.println("ERROR DE ARCHIVO:" + e);
                        }
                }

        protected void abreCargaImagen(){}


        protected void abreCargaRespuesta(){
                PMaestro_PreguntasPredefinidas dlg = new
PMaestro_PreguntasPredefinidas(this,SERVLET_PATH,servletURL,txtaAreaTexto,cveMa
estro,cveUEA);
                Dimension dlgSize = dlg.getPreferredSize();
                Dimension frmSize = getSize();
                Point loc = getLocation();
                dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height -
dlgSize.height) / 2 + loc.y);
                dlg.setModal(true);
                dlg.show();

        }


  void btnEnviar_actionPerformed(ActionEvent e) {
        imageControl4.setVisible(false);
  //    imageControl5.setVisible(false);
        imageControl3.setVisible(false);
                enviar();
        }

        void btnCargar_actionPerformed(ActionEvent e) {
                String lbl=chkboxgImgTxt.getSelectedCheckbox().getLabel();
                if(bRespondePreg==false){
                        if(lbl.equals("Texto")) abreCargaTxt();
```

```
                else abreCargaImagen();
            }
            else abreCargaRespuesta();
        }

        void btnResponder_actionPerformed(ActionEvent e) {
            String pregunta=null;
            pregunta=listaPreguntas.getSelectedItem();
            p("Select");
            if(pregunta==null)
                errorPregNoSelec();
            else{
                txtaAreaTexto.setText("");
                txtaAreaTexto.append("\nPREGUNTA DE  "+pregunta+"\n");
            }
            bRespondePreg=true;                    ////////////////*quitar cuando sea el
verdadero*/
        }


        void BGuardar_mouseClicked(MouseEvent e)
        {
             GuardaTxt();
        }


  void btnCargarHis_actionPerformed(ActionEvent e,List Lista2)
        {
            String nombre=null;
         nombre =listaUsuarios.getSelectedItem();


         System.out.println(nombre);
         if (nombre == null)
              error_Alumno_No_Selec();
         else     // aqui madar nombre al servlet para que busque en la base
                          // de datos  y llene una listaUsuarios o un vector para
mandarlo
                           // al cuadro de dialogo que lo mostrara en el area de texto
                          // de este mismo.  falta otro metodo de recepcion de datos
         //recibe la imformacion del servelt para la consulta
            MuestraHistorial(nombre,Lista2);


        }

 /*guarda en un archivo lo que se le ha enviado a los laumnos*/

 protected void GuardaTxt(){
                    Frame f=new Frame();
                    String linea;
                    String archivo;
```

```
                    FileDialog fd = new FileDialog(f,"GUARDAR
ARCHIVO",FileDialog.SAVE);
                    fd.show();
                    archivo= fd.getDirectory()+fd.getFile();          //CAMBIAR PARA
GUARDAR

                    linea=Area2.getText();
                    try{
                        FileOutputStream os = new FileOutputStream(archivo);
                        DataOutputStream ods=new DataOutputStream(os);
                        ods.writeUTF(linea);
                       ods.close();
                     }
                    catch(IOException e){
                      System.out.println("ERROR DE escritura ARCHIVO:" + e);
                    }


                }

        /*aqui recibo la cadena con la imformacion para desplegar
  en el area de texto*/                                          //quitar

 //Help | About action performed

        public void helpAbout_actionPerformed(ActionEvent e) {
                PMaestro_AboutBox dlg = new PMaestro_AboutBox(this);
                Dimension dlgSize = dlg.getPreferredSize();
                Dimension frmSize = getSize();
                Point loc = getLocation();
                dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height -
dlgSize.height) / 2 + loc.y);
                dlg.setModal(true);
                dlg.show();
        }

/*manda llamar la clase PMaestro_PregNoSelec para crear un cuadro de dialogo*/
        protected void errorPregNoSelec(){
                PMaestro_PregNoSelec dlg = new PMaestro_PregNoSelec(this);
                Dimension dlgSize = dlg.getPreferredSize();
                Dimension frmSize = getSize();
                Point loc = getLocation();
                dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height -
dlgSize.height) / 2 + loc.y);
                dlg.setModal(true);
                dlg.show();

        }


  protected void MuestraHistorial(String nombre,List Lista2)
        {               /**cambiar**/
```

```
        CuadroDialogo d = new CuadroDialogo(this,nombre,Lista2);
   Dimension dlgSize = d.getPreferredSize();
                Dimension frmSize = getSize();
                Point loc = getLocation();
                d.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height -
dlgSize.height) / 2 + loc.y);           d.setModal(true);
        d.show();
        }

  protected  void error_Alumno_No_Selec(){
        ErrAlumnoNoSel dlg_error= new ErrAlumnoNoSel(this);
   Dimension dlgSize =dlg_error.getPreferredSize();
        Dimension frmSize = getSize();
        Point loc = getLocation();
        dlg_error.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height -
dlgSize.height) / 2 + loc.y);
   dlg_error.setModal(true);
        dlg_error.show();
       }


  void this_windowClosed(WindowEvent e)
  {
       if(HiloColector!=null&&HiloColector.isAlive()){
                       HiloColector.stop();

      ///cuando cierra el chat se destruye el hilo colector

                }
   HiloColector=null;
                abandonar();
                System.exit(0);
   //HiloColector.destroy();
  }

}/*de todo*/



import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
import java.net.URL.*;
import java.io.*;
import com.sun.java.swing.*;
import com.sun.java.swing.border.*;

public class PMaestro_PreguntasPredefinidas extends Dialog implements ActionListener{
        String SERVLET_PATH;
        JPanel panel1 = new JPanel();
        JPanel panel2 = new JPanel();
```

```
    private void jbInit() throws Exception {
        //imageIcon = new ImageIcon(getClass().getResource("your image name
goes here"));
        this.setTitle("About");
        setResizable(false);
        panel1.setLayout(null);
        panel2.setLayout(borderLayout2);
        insetsPanel1.setLayout(flowLayout1);
        insetsPanel2.setLayout(flowLayout1);
        insetsPanel2.setBorder(new EmptyBorder(10, 10, 10, 10));
        gridLayout1.setRows(4);
        gridLayout1.setColumns(1);
        lstPreguntas.setBounds(new Rectangle(29, 93, 213, 303));
        txtaRespuesta.setBounds(new Rectangle(374, 93, 365, 300));
        btnAceptar.setBounds(new Rectangle(470, 419, 184, 43));
        btnAceptar.setLabel("Aceptar");
        btnAceptar.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                btnAceptar_actionPerformed(e);
            }
        });
        btnCancelar.setBounds(new Rectangle(86, 425, 93, 33));
        btnCancelar.setLabel("Cancelar");
        btnCancelar.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                btnCancelar_actionPerformed(e);
            }
        });
        btnCargarResp.setBounds(new Rectangle(254, 203, 112, 36));
        btnCargarResp.setLabel("Cargar Respuesta");
        btnCargarResp.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                btnCargarResp_actionPerformed(e);
            }
        });
        lblPregs.setBounds(new Rectangle(95, 62, 61, 24));
        lblPregs.setText("Preguntas");
        lblResp.setBounds(new Rectangle(539, 68, 70, 24));
        lblResp.setText("Respuesta");
        label2.setText(version);
        insetsPanel3.setLayout(gridLayout1);
        insetsPanel3.setBorder(new EmptyBorder(10, 60, 10, 10));
        button1.setText("OK");
        button1.addActionListener(this);
        insetsPanel2.add(imageControl1, null);
```

```
        void btnCargarResp_actionPerformed(ActionEvent e)
        {
                int index=lstPreguntas.getSelectedIndex();
                String texto=null;
                //texto= ctrlPregs.getPregunta(index);
                txtaRespuesta.append(texto);
        }

}

class CtrlPreguntas{
        String cveMaestro;
        String cveUEA;
        String SERVLET_PATH;
        String servletURL;

        CtrlPreguntas(String SERVLET_PATH,String servletURL,String cveMaestro,String
cveUEA){
                this.cveMaestro=cveMaestro;
                this.cveUEA=cveUEA;
                this.SERVLET_PATH=SERVLET_PATH;
                this.servletURL=servletURL;
        }
        protected Enumeration getPreguntas(){
                Vector pregs=null;
                String
query="?modo=lst_preguntas&cveUEA="+cveUEA+"&cveUsuario="+cveMaestro;
                query=SERVLET_PATH+query;
                try{
                        URL urlServlet=new URL(servletURL);
                        URL urlServletGet=new URL(urlServlet,query);
                        URLConnection conexion=  urlServletGet.openConnection();
                        conexion.setDefaultUseCaches(false);
                        conexion.setUseCaches(false);
                        conexion.setDoInput(true);
                        conexion.setDoOutput(false);
                        ObjectInputStream                                       in=new
ObjectInputStream(conexion.getInputStream());
                        pregs=(Vector)in.readObject();
                }
                catch(IOException e1){
                        e1.printStackTrace();
                }
                catch(ClassNotFoundException e3){
                        e3.printStackTrace();
                }
                Enumeration enum=pregs.elements();
                return  enum;

        }
```

```
        protected String getRespuesta(String cvePregunta){
                String sigLinea=null;
                String query="?modo=repuesta&cvePregunta="+cvePregunta;
                query=SERVLET_PATH+query;
                try{
                        URL urlServlet=new URL(servletURL);
                        URL urlServletGet=new URL(urlServlet,query);
                        URLConnection conexion=  urlServletGet.openConnection();
                        conexion.setDefaultUseCaches(false);
                        conexion.setUseCaches(false);
                        conexion.setDoInput(true);
                        conexion.setDoOutput(false);
                        conexion.connect();
                        BufferedReader in=new BufferedReader(new
InputStreamReader(conexion.getInputStream()));
                        sigLinea=in.readLine();
                        if(sigLinea.startsWith("+")){
                                sigLinea=in.readLine();
                                return sigLinea;
                        }
                        else sigLinea="Error al cargar respuesta"+sigLinea;
                }
                catch(IOException e1){
                        e1.printStackTrace();
                }
                //Enumeration enum=pregs.elements();
                return  sigLinea;


        }


}

import borland.jbcl.*;
import java.awt.*;
import java.util.*;
import java.awt.event.*;
import borland.jbcl.control.*;
import borland.jbcl.layout.*;
import com.sun.java.swing.*;

public class CuadroDialogo extends Dialog {
 Button BtnCerrar=new Button();
  public CuadroDialogo(Frame padre,String nombre,List Lista2) {

        super(padre,"Historial de "+ nombre, false);
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try{
          cuadro();
          dibujaArea(Lista2);
          dibujaBoton();
```

```
        }
        catch(Exception e){
         e.printStackTrace();
        }
   pack();
 }
        private void cuadro() throws Exception{
          this.setBackground(new Color(129, 181, 172));
          this.setLayout(null);
          setLocation(450,15);
          resize(340,550);
        }
        private void dibujaArea(List Lista2) throws Exception{
        TextAreaControl Area= new TextAreaControl();
        //Area.
        Area.setBackground(new Color(255, 255, 255));
        Area.setBounds(new Rectangle(15,25,310,450));
        Area.appendText(Lista2.getSelectedItem());
        Area.setEditable(false);
        this.add(Area);

 }
private void dibujaBoton() throws Exception{
        BtnCerrar.setBounds(new Rectangle(105,500,135,32));
        BtnCerrar.setLabel("Cerrar");
        BtnCerrar.addActionListener(new java.awt.event.ActionListener()
        {
          public void actionPerformed(ActionEvent e){
                                BtnCerrar_ActionPerformed(e);
          }
        });
        this.add(BtnCerrar);
 }
 //cierra el cuadro de dialogo con el icono " X "
        protected void processWindowEvent(WindowEvent evt){
         if (evt.getID()== WindowEvent.WINDOW_CLOSING){
                this.dispose();
                 }
         super.processWindowEvent(evt);
 }
 //cierra el cuadro de dialogo
 public void BtnCerrar_ActionPerformed(ActionEvent e){
         dispose();
 }

}
```

CHAT APPLET

```
import java.applet.Applet;
import java.awt.*;
import java.net.*;
```

```
import java.io.*;
import java.util.*;

public class ChatApplet extends Applet implements Runnable{

   private static final String SERVLET_PATH = "/servlet/chat.ChatServlet";

   Label userInfo;
   Label messageInfo;
   Label listInfo;
   Button sendButton;
   TextField userText;
   TextArea  messageText;
   List userList;
   URL chatURL;
   URL servletURL;
   URLConnection connect;
   volatile private boolean loggedin = false;
   String username=null;
   Thread pollingThread = null;
   //Lista de parametros.
   public String[][] getParameterInfo() {
      return null;
   }
   //Describe el applet.
   public String getAppletInfo() {
         return "ChatApplet - Applet";
   }

   public synchronized void init() {
      if (pollingThread != null) return;  //Llamara a init() siempre antes de hilar
      super.init();
      resize(500,300);
      userInfo = new Label("Teclea mensaje:");
      messageInfo = new Label("Area de mensajes:");
      userText = new TextField(40);
      sendButton = new Button("Enviar");
      messageText = new TextArea(10,40);
      messageText.setEditable(false);
      //Inicializa el panel principal
      Panel mainp = new Panel();
      GridBagLayout gbl = new GridBagLayout();
      GridBagConstraints gbc = new GridBagConstraints();
      gbc.weightx = 0;
      gbc.weighty = 0;
      gbc.gridx = 0;
      gbc.gridy = 0;
      gbc.gridwidth = 10;
      gbc.gridheight = 1;
      gbc.anchor = GridBagConstraints.CENTER;
      gbc.fill = GridBagConstraints.NONE;
      mainp.setLayout(gbl);
```

```
        gbl.setConstraints(userInfo, gbc);
        mainp.add(userInfo);
        gbc.gridy = 1;
        gbc.gridwidth = 9;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbl.setConstraints(userText, gbc);
        mainp.add(userText);
        gbc.gridx = 9;
        gbc.gridwidth = 1;
        gbc.fill = GridBagConstraints.NONE;
        gbl.setConstraints(sendButton, gbc);
        mainp.add(sendButton);
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 10;
        gbl.setConstraints(messageInfo, gbc);
        mainp.add(messageInfo);
        gbc.gridy = 3;
        gbc.weighty = 100;
        gbc.gridheight = 10;
        gbc.fill = GridBagConstraints.BOTH;
        gbl.setConstraints(messageText, gbc);
        mainp.add(messageText);

        Panel userp = new Panel();
        userp.setLayout(new BorderLayout());
        listInfo = new Label("Usuarios en el chat:");
        userp.add("North",listInfo);
        userList = new List(10, false);
        userList.addItem("Ningun usuario.");
        userp.add("Center",userList);


        setLayout(new BorderLayout());
        add("Center",mainp);
        add("East",userp);
        // URL base(Donde se cargo el applet.
        chatURL = getCodeBase();
        //Inicializa y comienza el hilo colector.
        int currPriority = Thread.currentThread().getPriority();
        int newPriority = currPriority == Thread.MIN_PRIORITY ? Thread.MIN_PRIORITY :
currPriority - 1;
        pollingThread = new Thread(this,"Chat colector");
        pollingThread.setDaemon(true);
        pollingThread.setPriority(newPriority);
        pollingThread.start();
        p("Chat: Iniciando coleccion");
        username=getParameter("nickname");
        System.out.println("us: "+username);
        login();
```

```
}
//Inicializa o reanuda el hilo
/**
 * Start polling thread if not already running.
 */
public synchronized void start() {
   if (!isLoggedin() && username != null) {
      //login();
   }
   if (pollingThread != null && pollingThread.isAlive()) {
      pollingThread.resume();
      p("Chat: reanudando hilo");
   } else {
      p1("Chat: No hay hilo colector!");
      pollingThread = new Thread(this,"Chat colector");
      pollingThread.setDaemon(true);
      pollingThread.start();
      p("Chat: iniciando coleccion");
   }
}

/**
 *
 */
public synchronized void stop() {
   if (pollingThread.isAlive()) {
      pollingThread.suspend();
      p("Chat: suspendiendo coleccion");
   } else {
      p1("Chat: hilo principal muerto y en stop()!");
   }
   logout();
}

public synchronized void destroy() {
   if (pollingThread != null && pollingThread.isAlive()) {
      pollingThread.stop();
      pollingThread = null; //En caso de regresar
      p("Chat: parando coleccion");
   }
   logout();
}

public void run() {

   p("Chat:  iniciando run()");
   while (!Thread.interrupted()) {
      if (isLoggedin()) {
         pollList();
         poll();
         p("Chat: colectando");
      } else {
```

```
            pollList();
            p("Chat: no ha entrado al chat para coleccion");
         }
         try {
            Thread.sleep(5000); // Dormir por 1 seg    //Aqui lo cambie
         } catch (InterruptedException e) {}
      }
      p("Chat: saliendo de run()");
   }


   private void login() {
      if (username == null) return;
            String queryString = SERVLET_PATH +
"?modo=nuevo_usuario&usuario="+URLEncoder.encode(username);
      p("Tratando de entrar como: "+username);
      try {
         connect = (new URL(chatURL,queryString)).openConnection();
         connect.setDefaultUseCaches(false); //Para futuras conexiones
         connect.setUseCaches(false); //No usar caches para acelerar envio
         connect.setDoInput(true);
         connect.setDoOutput(false);
         connect.connect();
         p("Realizando conexion con: "+connect);
         BufferedReader in = new BufferedReader(new
InputStreamReader(connect.getInputStream()));
         String response = in.readLine();
         if (response.startsWith("+")) {
            setLoggedin(true);
            showStatus("Entro al chat como " + username);
            p("Entro como "+username);
            userInfo.setText("Teclea mensaje: ");
            repaint();
         } else {
            showStatus("Error al entrar" + response);
            p("No pudo entrar como "+username);
            System.err.println("Error al intentar conexion:" +response);

         }
      } catch (MalformedURLException e2) {
         System.err.println(e2);
         e2.printStackTrace(System.err);
         showStatus("Error al tratar de conectarse");
      } catch (IOException e1) {
         System.err.println(e1);
         e1.printStackTrace(System.err);
         showStatus("Error al tratar de conectarse");
      }
   }

   private void logout() {
      if (!isLoggedin() || username == null) return;
```

```
            String queryString = SERVLET_PATH +
"?modo=borra_usuario&usuario="+URLEncoder.encode(username);
      try {
         connect = (new URL(chatURL,queryString)).openConnection();
         connect.setUseCaches(false);
         connect.setDoInput(true);
         connect.setDoOutput(false);
         connect.connect();
         BufferedReader in = new BufferedReader(new
InputStreamReader(connect.getInputStream()));
         String response = in.readLine();
         if (response.startsWith("+")) {
            setLoggedin(false);
            showStatus("Usuario" + username + "ha salido del chat");
         } else {
            showStatus("Error al salir" + response);
            System.err.println("Error al salir" +response);
         }
      } catch (MalformedURLException e2) {
         System.err.println(e2);
         e2.printStackTrace(System.err);
         showStatus("Error al intentar salir");
      } catch (IOException e1) {
         System.err.println(e1);
         e1.printStackTrace(System.err);
         showStatus("Error al salir");
      }
   }

   private void send() {
      if(!isLoggedin()){return;}
      String message = userText.getText();
      if (message.equals("")) return;  //No enviar un mensaje vacio
      userText.setText("");
      showStatus("Sending message");
       String queryString = SERVLET_PATH +
"?modo=envia&usuario="+URLEncoder.encode(username);
      queryString = queryString +"&mensaje="+URLEncoder.encode(message);
      try {
         connect = (new URL(chatURL,queryString)).openConnection();
         connect.setUseCaches(false); //Para este
         connect.setDoInput(true);
         connect.setDoOutput(false);
         connect.connect();
         BufferedReader in = new BufferedReader(new
InputStreamReader(connect.getInputStream()));
         String response = in.readLine();
         if (response.startsWith("+")) {
            showStatus("Mensaje enviado");
         } else {
            showStatus("Error al enviar mensaje" + response);
            System.err.println("Error al enviar mensaje" + response);
```

```
            }
        } catch (MalformedURLException e2) {
            System.err.println(e2);
            e2.printStackTrace(System.err);
            showStatus("Error al enviar mensaje");
        } catch (IOException e1) {
            System.err.println(e1);
            e1.printStackTrace(System.err);
            showStatus("Error al enviar mensaje");
        }
    }


    private void poll() {
            String queryString = SERVLET_PATH +
"?modo=colecta&usuario="+URLEncoder.encode(username);
        try {
            connect=(new URL(chatURL,queryString)).openConnection();
            connect.setUseCaches(false);
            connect.setDefaultUseCaches(false);
            ObjectInputStream in=new ObjectInputStream(connect.getInputStream());
            String nextLine = (String)in.readObject();
            if (!nextLine.startsWith("+")) {
                showStatus("Error obteniendo mensajes del servidor");
                System.out.println("Error obteniendo mensajes : "+nextLine);
                return;
            }
            nextLine = (String)in.readObject();
            while (nextLine != null && !nextLine.equals(".")) {
                System.out.println("Linea:"+nextLine);
                messageText.append(nextLine+"\r\n");
                repaint();
                nextLine = (String)in.readObject();
            }
        }
        catch (IOException e) {
            System.err.println(e);
            e.printStackTrace(System.err);
            showStatus("Error al checar el mensaje");
        }
        catch(ClassNotFoundException e1){
            System.err.println(e1);
            e1.printStackTrace(System.err);
            showStatus("Error al checar el mensaje OBJ");

        }

    }
    //Pregunta por la lista de usuarios
    private void pollList() {
            String queryString = SERVLET_PATH + "?modo=lista";
      Vector users = new Vector();
```

```
    try {
            System.out.println("chatURL: " + chatURL);
            System.out.println("queryString: " + queryString);
      URL listURL = new URL(chatURL,queryString);
            System.out.println("listURL: " + listURL);
      URLConnection listConn = listURL.openConnection();
      listConn.setDefaultUseCaches(false);
      listConn.setUseCaches(false);
      listConn.connect();
      BufferedReader in = new BufferedReader(new
InputStreamReader(listConn.getInputStream()));
      String nextLine = in.readLine();
      if (!nextLine.startsWith("+")) {
          showStatus("Error al obtener lista de usuarios del servidor");
          p1("Error obteniendo lista de usuarios del servidor");
          return;
      }
      nextLine = in.readLine();
      while (nextLine != null && !nextLine.equals(".")) {
          p("Read user: "+nextLine);
          users.addElement(nextLine);
          nextLine = in.readLine();
      }
        if (!users.isEmpty()) {
          userList.removeAll();
          int size = users.size();
          for (int i = 0; i < size; i++) {
             userList.addItem((String)users.elementAt(i));
          }
        } else {
          userList.removeAll();
          userList.addItem("Ningun usuario dentro");
        }
        repaint();

      } catch (IOException e) {
        System.err.println(e);
        e.printStackTrace(System.err);
        showStatus("Error al obtener lista"+e);
      }

  }
  //Checa si el usuario esta dentro.
  public boolean isLoggedin() {
    return loggedin;
  }
  //Pone variable
  protected void setLoggedin(boolean newval) {
    loggedin = newval;
  }

  public boolean action(Event evt, Object arg) {
```

```
        if (evt.target == sendButton || evt.target == userText) {
            if (isLoggedin())
                send();
            else {
                username = userText.getText();
                if (username.length() > 10) {
                    showStatus("10 o menos caracteres!");
                } else {
                    userText.setText("");
                    login();
                }
            }
            return true;
        }

        return super.action(evt,arg);

    }

    private void p(String debug) {
        System.err.println("Chat:"+debug);
    }

    private void p1(String debug) {
        System.err.println("Chat:"+debug);
    }

    private String getString(String q) {
            return  "/servlet/chat.ChatServlet" + "?" + q;
    }


    public ChatApplet() {
      try  {
        jbInit();
      }
      catch (Exception e) {
        e.printStackTrace();
      }
    }

    private void jbInit() throws Exception {
      this.setBackground(new Color(129, 181, 180));
    }
}
```

CHAT SERVLET

```
package chat;
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
```

```
import java.util.*;

public class ChatServlet extends HttpServlet implements Runnable {

   private ListaUsuarios usuarios = new ListaUsuarios();
   private final static String paginaDefault = "/public_html/ChatApplet.html";
   private Thread recolector;
   // Timeout
   protected static final int TIMEOUT = 95 * 60 * 1000;//5 minutos

   public void init(ServletConfig config) throws ServletException {
      super.init(config);
      int currPriority = Thread.currentThread().getPriority();
      int newPriority = currPriority == Thread.MIN_PRIORITY ? Thread.MIN_PRIORITY :
currPriority - 1;
      recolector = new Thread(this,"Chat Recolector");
      recolector.setDaemon(true);
      recolector.setPriority(newPriority);
      recolector.start();
      return;
   }

   public void destroy() {
      super.destroy();
      if (recolector != null && recolector.isAlive()) {
         recolector.stop();
      }
      return;
   }

   public void doGet(HttpServletRequest req, HttpServletResponse res)
     throws IOException, ServletException {
      Administracion admon=new Administracion(res,req);
      String usuario, modo ;
      if (req.getQueryString() == null) {  //Regresar a la pagina inicial...
         if (getInitParameter("startpage" )== null)
            res.sendRedirect(paginaDefault);
         res.sendRedirect(getInitParameter("startpage"));
         return;
      }
      modo = req.getParameter("modo");
      usuario = req.getParameter("usuario");
      if(usuarios.existe("maestro")||usuario.equals("maestro")){

         if (modo == null) {
            admon.enviaError("Modo no iniciado");
            return;
         }
         else if (usuario == null && !modo.equals("lista")) {
            admon.enviaError("Usuario no indicado");
            return;
         }
```

```
        else if (modo.equals("nuevo_usuario"))
          admon.addNuevo(usuario,usuarios);
        else if (modo.equals("borra_usuario"))
          admon.borraUsuario(usuario,usuarios);
        else if (modo.equals("colecta"))
          admon.colectaMensajes(usuario,usuarios);
        else if (modo.equals("envia"))
          admon.envia(usuario,usuarios);
        else if (modo.equals("lista"))
          admon.regresa_listaUsuarios(usuarios);
        else
          admon.enviaError("Modo invalido.");
      }
       else
        admon.enviaError("Clase no iniciada");
  }

  public void run() {
    while (!Thread.interrupted()) {
      Enumeration inactive = usuarios.inactive(TIMEOUT);
      while (inactive.hasMoreElements()) {
        String usuario = (String)inactive.nextElement();
        usuarios.borra(usuario);
        usuarios.addMensaje(new Mensaje("Chat",usuario + " ha finalizado su
Timeout."));
        usuarios.addMensaje(new Mensaje("Chat","Desechando "+usuario+" del
chat."));
        log("Usuario "+usuario+" fuera de tiempo.  Desechado del chat.");
      }
      try {
        Thread.sleep(TIMEOUT); // Dormir por el TIMEOUT
      } catch (InterruptedException e) {}
    }
  }

  public void doPost(HttpServletRequest req,HttpServletResponse res)throws
IOException,ServletException{
    Administracion admon=new Administracion(res,req);
    admon.envia(usuarios);
  }


}//Fin servlet
/**************************************************************************
  Clases básicas
**************************************************************************/
class Administracion{
  HttpServletResponse res=null;
  HttpServletRequest req=null;

  Administracion(HttpServletResponse res,HttpServletRequest req){
    this.res=res;
```

```
  this.req=req;
}

protected void addNuevo(String usuario,ListaUsuarios usuarios){
  if (usuarios.existe(usuario)) {
        enviaError("Usuario existe");
        return;
      }
  if (!usuarios.add(usuario)) {
    enviaError("Problema al añadir usuario");
    return;
  }
  usuarios.addMensaje(new Mensaje("Aula Virtual: ",usuario + " ha entrado."));
  enviaResp("Usuario ha entrado al Aula Virtual.");

}

protected void borraUsuario(String usuario,ListaUsuarios usuarios){
  if (!usuarios.existe(usuario)){
    enviaError("Usuario no existe");
    return;
  }
  else if (!usuarios.borra(usuario)){
        enviaError("Problema al borrar al usuario");
        return;
  }
  usuarios.addMensaje(new Mensaje("Chat",usuario + " ha dejado el chat."));
  enviaResp("Usuario ha salido.");

}

protected void colectaMensajes(String usuario,ListaUsuarios usuarios){
  if (!usuarios.existe(usuario)){
    enviaError("Invalido usuario");
    return;
  }
  //Para maestro
  if(usuario.equals("maestro")) regresaPreguntas( usuarios.getMensajes("maestro"));
  //Para alumno
  else regresaMensajes(usuarios.getMensajes(usuario));
  usuarios.resetUsuario(usuario);

}
//Alumno envia pregunta
protected void envia(String usuario,ListaUsuarios usuarios){
  String msjusuario = req.getParameter("mensaje");
  if (msjusuario == null) {
    enviaError("Mensaje no inicializado para envio");
    return;
  }
  usuarios.addPregunta(new Mensaje(usuario,msjusuario));
  enviaResp("Mensaje aceptado.");
```

```
}
//Maestro envia mensaje
protected void envia(ListaUsuarios usuarios){
  try{
    String mensaje=null;
    ObjectInputStream in=new ObjectInputStream(req.getInputStream());
    mensaje=(String)in.readObject();
    if (mensaje == null) {
      enviaError("Mensaje no inicializado para envio");
      return;
    }
    usuarios.addMensaje(new Mensaje("maestro",mensaje));
    enviaResp("Mensaje aceptado.");
  }
  catch(IOException e){
    e.printStackTrace();
  }
  catch(ClassNotFoundException e1){
    e1.printStackTrace();
  }
}

protected void enviaError(String error) {
    try {
       PrintWriter pout = new PrintWriter(res.getOutputStream());
       pout.print("-ERR "+error+"\r\n");
       pout.flush();
       pout.close();
    } catch (IOException e) {
       // Pendiente
    }
 }

 protected void enviaResp(String resp) {
    try {
       res.setDateHeader("Expira",System.currentTimeMillis());
       PrintWriter pout = new PrintWriter(res.getOutputStream());
       pout.print("+OK "+resp+"\r\n");
       pout.flush();
       pout.close();
    }
    catch (IOException e) {
       // Pendiente
    }
 }

 protected void regresaPreguntas(Enumeration lista_mensajes) {
    try {
       res.setDateHeader("Expira",System.currentTimeMillis());
       PrintWriter pout = new PrintWriter(res.getOutputStream());
       pout.print("+OK\r\n");
```

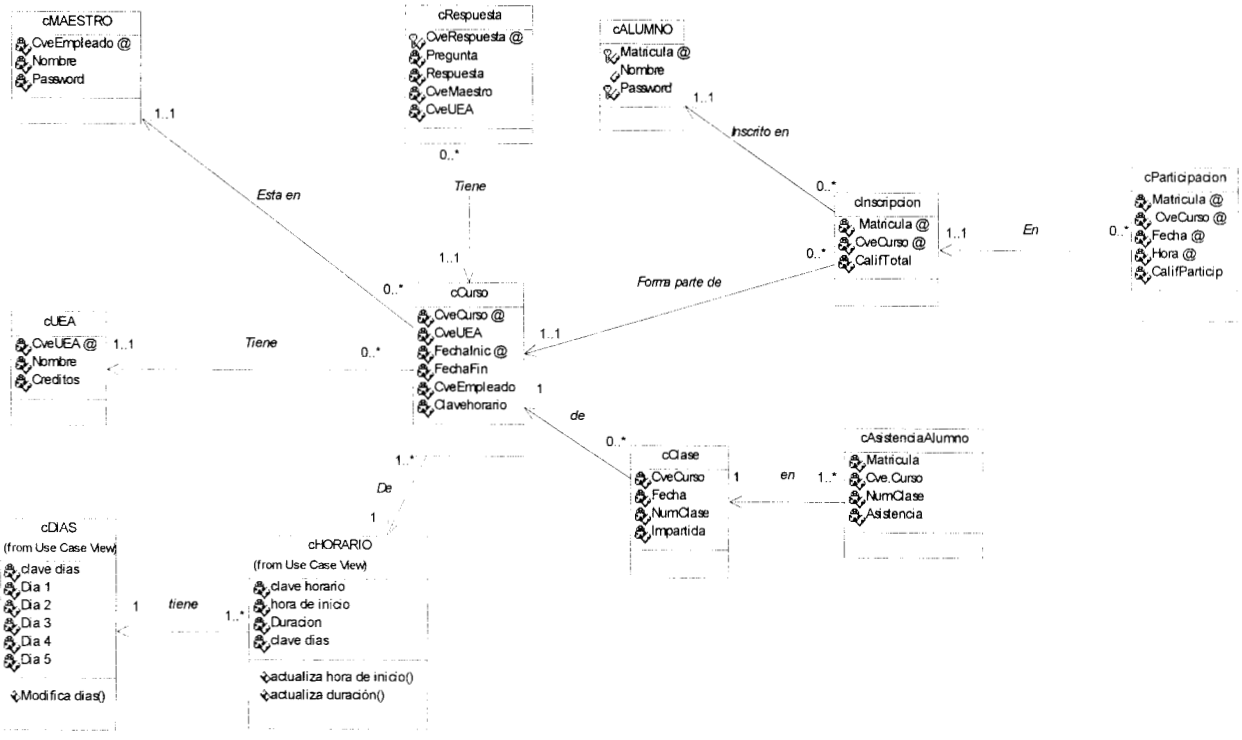
```
            while (lista_mensajes.hasMoreElements()) {
                Mensaje mensaje = (Mensaje) lista_mensajes.nextElement();
                pout.print(mensaje+"\r\n");
            }
            pout.print(".\r\n");
            pout.flush();
            pout.close();
        }
        catch (IOException e) {
            // Pendiente
        }
    }

    protected void regresaMensajes(Enumeration lista_mensajes){
     try{
        res.setDateHeader("Expira",System.currentTimeMillis());
        ObjectOutputStream out=new ObjectOutputStream(res.getOutputStream());
        out.writeObject("+OK\r\n");
        while (lista_mensajes.hasMoreElements()) {
             Mensaje mensaje = (Mensaje) lista_mensajes.nextElement();
             out.writeObject(mensaje+"\r\n");
        }
        out.writeObject(".\r\n");
        out.flush();
        out.close();
     }
     catch(IOException e){
        e.printStackTrace();
     }
    }
    protected void regresa_listaUsuarios(ListaUsuarios usuarios) {
     try {
        res.setDateHeader("Expira",System.currentTimeMillis());
        PrintWriter pout = new PrintWriter(res.getOutputStream());
        pout.print("+OK\r\n");
        Enumeration listaUsuarios = usuarios.lista();
        while (listaUsuarios.hasMoreElements()) {
            pout.print((String)listaUsuarios.nextElement());
            pout.print("\r\n");
        }
        pout.print(".\r\n");
        pout.flush();
     }
     catch (IOException e) {
        //Pendiente
     }
    }
}//Fin administración

//Lista de usuarios y sus operaciones.
class ListaUsuarios {
```

```
    private Hashtable lista = new Hashtable();
    private Hashtable active = new Hashtable();

    ListaUsuarios() {
    }

    protected synchronized boolean add(String usuario) {
        if (existe(usuario)) return false;
        lista.put(usuario, new ColaMensajes());
        touch(usuario);
        return true;
    }

    protected synchronized Enumeration lista() {
        return lista.keys();
    }

    protected synchronized boolean borra(String usuario) {
        if (!existe(usuario)) return false;
        lista.remove(usuario);
        active.remove(usuario);
        return true;
    }
//Checa la existencia de un usuario
    protected boolean existe(String usuario) {
        return lista.containsKey(usuario);
    }

    protected void addMensaje(Mensaje mensaje) {
        Enumeration todosUsuarios = lista.keys();
        while (todosUsuarios.hasMoreElements()){
            String usua=(String) todosUsuarios.nextElement();
            if(!usua.equals("maestro"))
              addMensajeUsuario(usua,mensaje);
        }
    }

   protected void addPregunta(Mensaje mensaje){
        addMensajeUsuario("maestro",mensaje);
    }

    private void addMensajeUsuario(String usuario, Mensaje mensaje) {

        ((ColaMensajes)lista.get(usuario)).add(mensaje);
    }

    protected void resetUsuario(String usuario) {
        ((ColaMensajes)lista.get(usuario)).reset();
    }
//Regresa una enumeracion de los mensajes esperados por el usuario
    protected Enumeration getMensajes(String usuario) {
        touch(usuario);
```

```
        return ((ColaMensajes)lista.get(usuario)).listaMensajes();
    }
    //Actualiza el Timeout para un usuario
    protected synchronized void touch(String usuario) {
        if (existe(usuario)) {
            active.put(usuario, new Long(System.currentTimeMillis()));
        }
    }
    //Regresa una enumeracion de los usuarios inactivos
    protected synchronized Enumeration inactive(int timeout) {
        Vector inactive = new Vector();
        long now = System.currentTimeMillis();
        long when = now - timeout;

        Enumeration usuarios = lista.keys();
        while (usuarios.hasMoreElements()) {
            String usuario = (String)usuarios.nextElement();
            if (((Long)active.get(usuario)).longValue() < when) {
                inactive.addElement(usuario);
            }
        }
        return inactive.elements();
    }
}//Fin ListaUsuarios

//Lista de mensajes
class ColaMensajes {

    private Vector cola_mensaje = new Vector();

    ColaMensajes() {}
    //Anade mensajes a la cola
    protected synchronized void add(Mensaje mensaje) {
        cola_mensaje.addElement(mensaje);
        notifyAll();
    }
     //Regresa una lista de mensajes en la cola y se bloquea hasta que un mensaje
     //esta listo para envio
    protected synchronized Enumeration listaMensajes() {
        try {
            if (cola_mensaje.isEmpty()) {
                wait(30*1000); //30 second timeout
            }
        } catch (InterruptedException ie) {
            //Pendiente
        }
        return cola_mensaje.elements();
    }

    protected synchronized void reset() {
         cola_mensaje.removeAllElements();
    }
```

```
}//Fin ColaMensajes

//Mensaje que envia un usuario y que guarda en la cola de todos los usuarios
class Mensaje {

   private String usuario;
   private String mensaje;

   Mensaje(String usuario, String mensaje) {
      this.usuario = usuario;
      this.mensaje = mensaje;
   }

   public String toString() {
      return usuario+": "+mensaje;
   }

   protected String getUser() {
      return usuario;
   }

   protected String getMensaje() {
      return mensaje;
   }
}//Fin Mensaje
```

MODELO DEL DOMINIO DEL PROBLEMA GENERAL

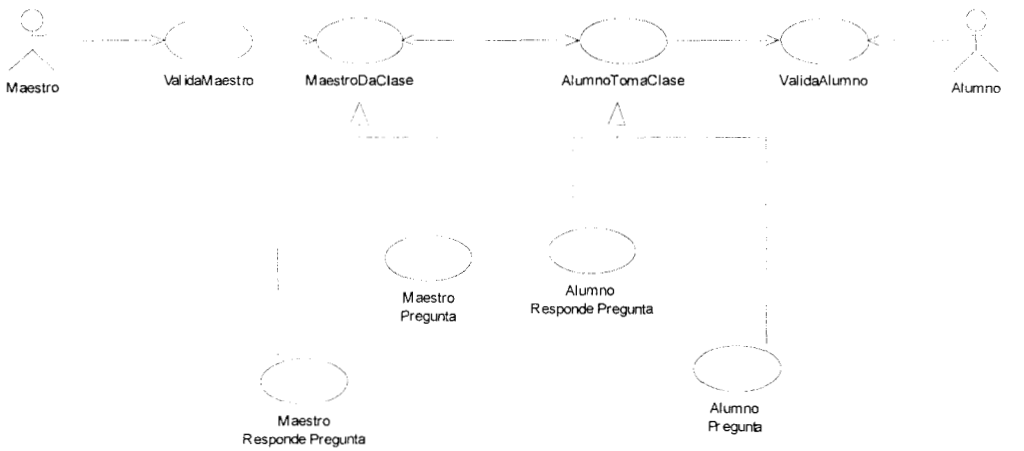


5. DESARROLLO PRACTICO (Segundo Prototipo).

En la etapa práctica del proyecto, el análisis se hizo empleando las técnicas del Análisis y Diseño Orientado a Objetos (ADOO), por lo que primeramente mostraremos un modelo de requerimientos general y después centraremos la atención en los casos individuales y sus especializaciones. En cada caso se presentara el modelo de casos, de interfaz y de dominio del problema.

5.1 Modelo de requerimientos a nivel general.

Diagrama General de Use Cases del Chat



5.2 Modelo de requerimientos para cada Use Case.

En esta parte analizaremos cada UC del Modelo de requerimientos a nivel general, la forma de hacerlo será primeramente el lado de los UC del Maestro y después continuaremos con los UC del Alumno.

5.2.1 Modelo de Requerimientos: Use Case ValidaMaestro

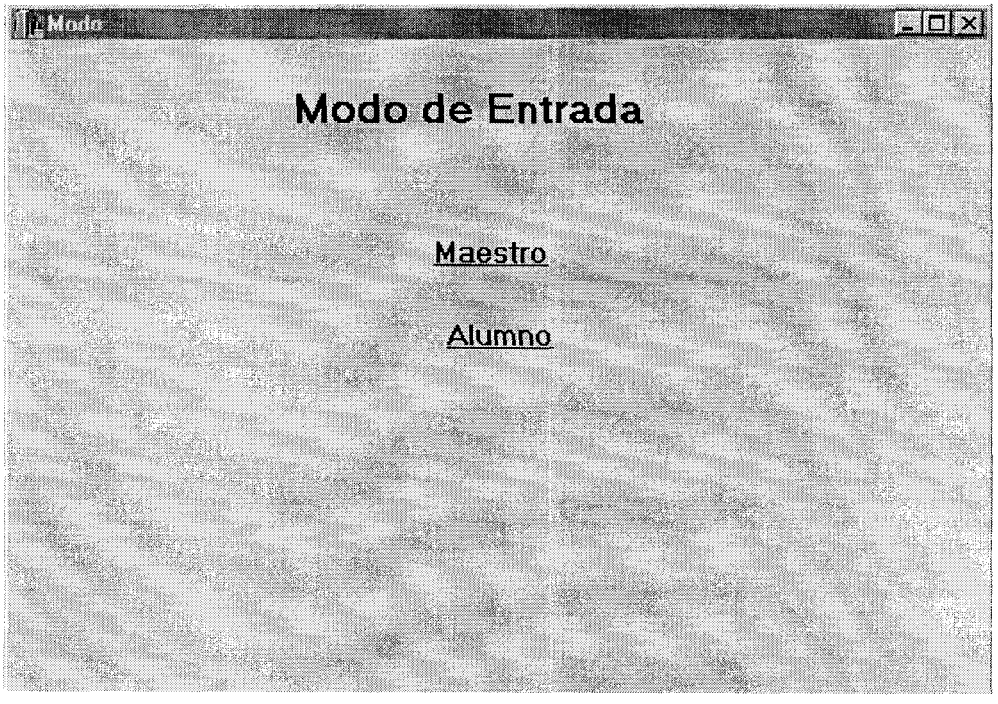
Modelo de Casos.

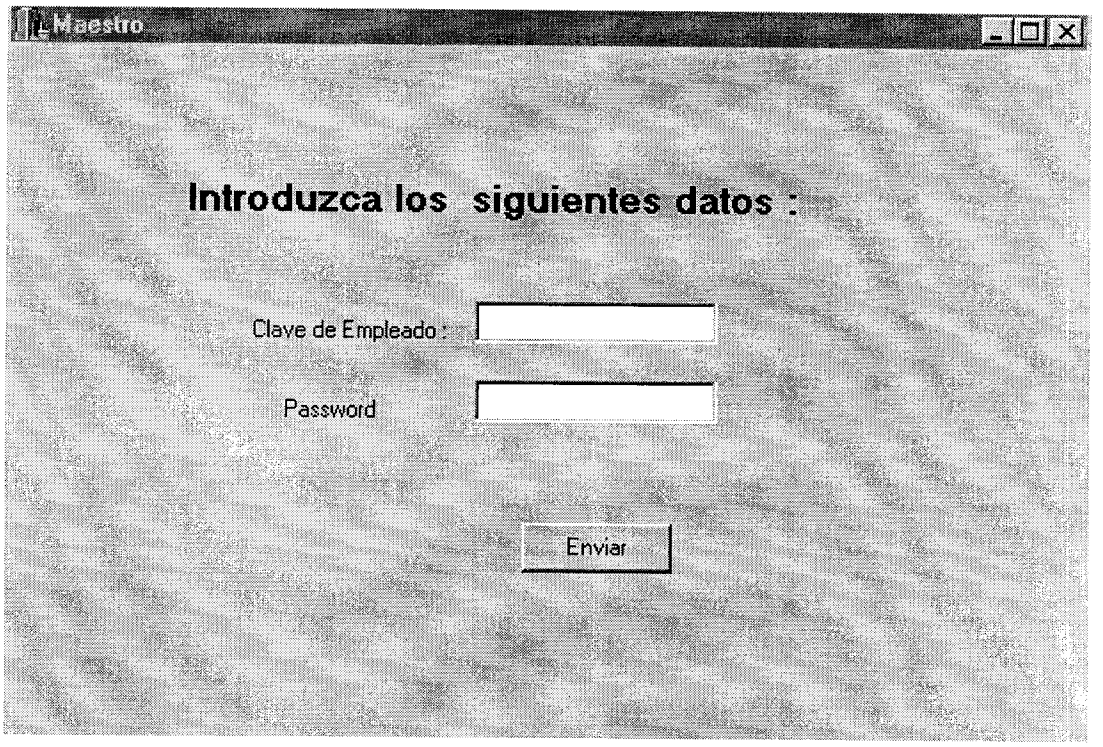
El siguiente diagrama de casos muestra la validación que tendrá que realizar el maestro, cada vez que ingrese al sistema.

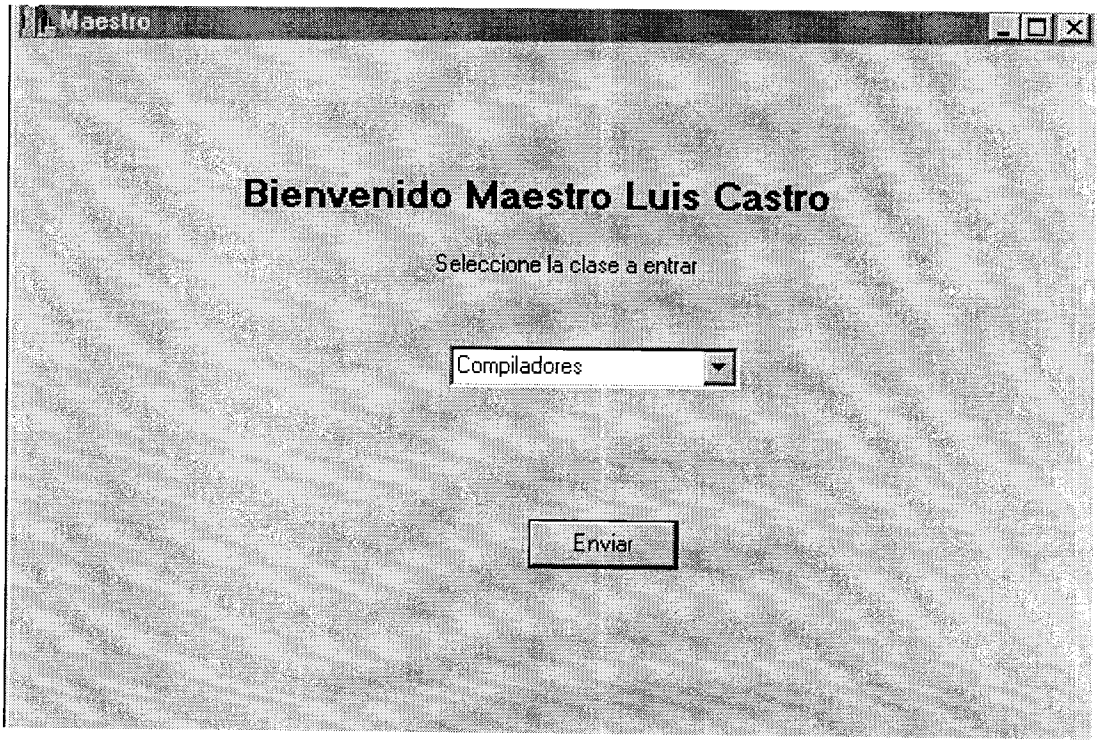
Descripción del caso:

Paso	Descripción UC Valida Maestro
1	El sistema despliega una lista de modo de entrada.
2	El maestro selecciona el modo de entrada maestro.
3	El sistema despliega una pantalla donde el maestro introduce su Clave de empleado, Password y presiona enviar.
4	El sistema valida la clave de empleado y su password.
5	Si los datos son correctos el sistema despliega una lista de cursos donde el maestro esta dado de alta para impartir clases.
6	El maestro elige el curso al que desea ingresar y presiona enviar.
7	El maestro entra al Aula Virtual.
8	Fin del Caso.

Modelo de Interfaz.




Maestro
Introduzca los siguientes datos :
Clave de Empleado :
Password
Enviar


Maestro
Bienvenido Maestro Luis Castro
Seleccione la clase a entrar
Compiladores
Enviar

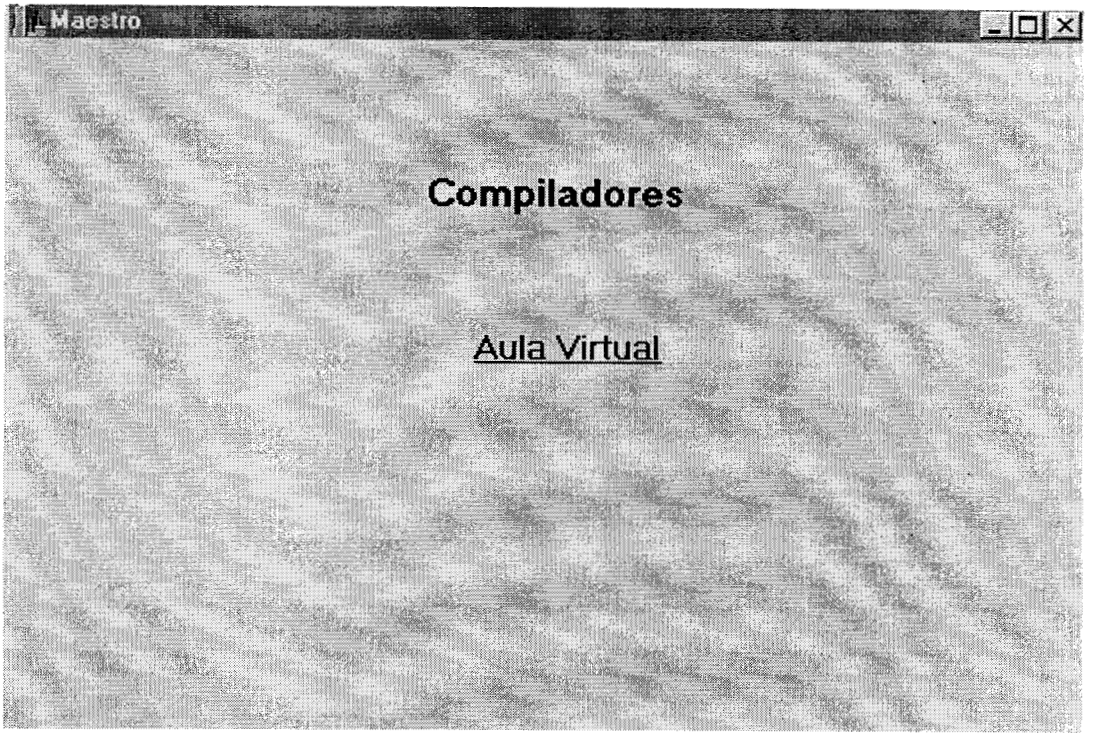
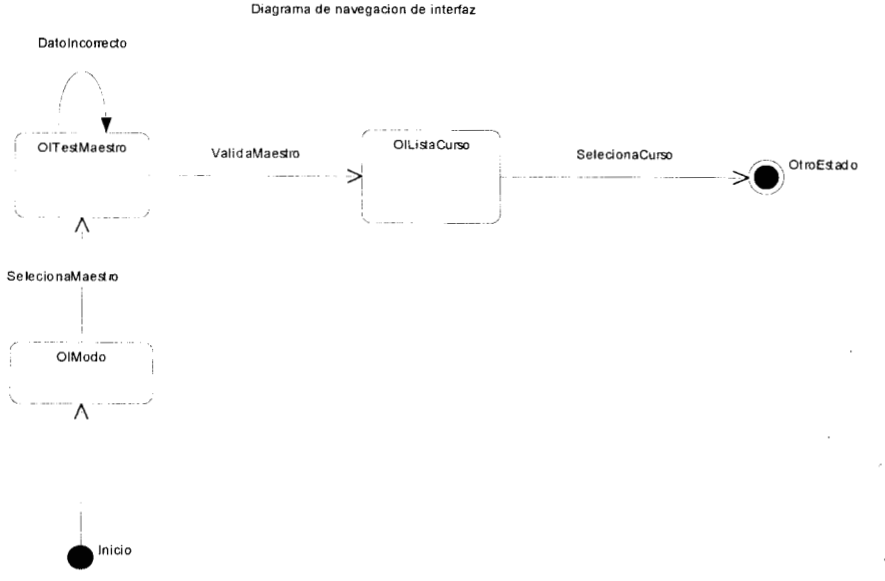
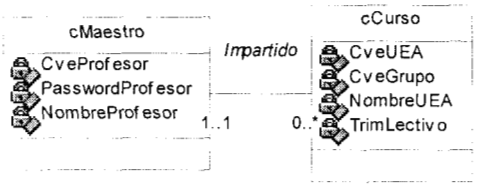


Diagrama de Navegación de Pantallas



Modelo del Dominio del Problema.

Paso	Descripción UC Valida Maestro
1	El **sistema** *despliega* una **lista** de **modo de entrada**.
2	El **maestro** *selecciona* el **modo de entrada** maestro.
3	El **sistema** *despliega* una **pantalla** donde el **maestro** *introduce* su **Clave de empleado**, **Password** y *presiona* enviar.
4	El **sistema** *valida* la **clave de empleado** y su **password**.
5	Si los **datos** son correctos el **sistema** *despliega* una **lista de cursos** donde el **maestro** esta dado de alta para *impartir* **clases.**
6	El **maestro** *elige* el **curso** al que *desea ingresar* y *presiona enviar.*
7	El **maestro** entra al **Aula Virtual**.
8	Fin del Caso.



5.2.2. Modelo de Requerimientos: Use Case MaestroDaClases

Este Use Case tiene dos especializaciones, que son:
- Use Case Maestro Pregunta
- Use Case Maestro Responde Pregunta

Estos Uses Cases se describen a continuación.

5.2.2.1. Modelo de Requerimientos: Use Case Maestro Pregunta

Modelo de Casos.

El siguiente diagrama de casos nos muestra como el maestro puede enviar una pregunta a los alumnos.

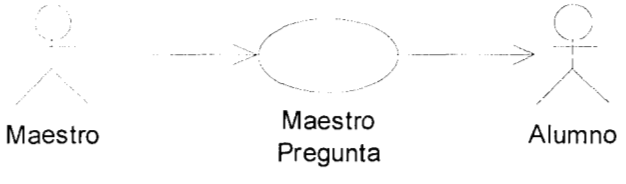
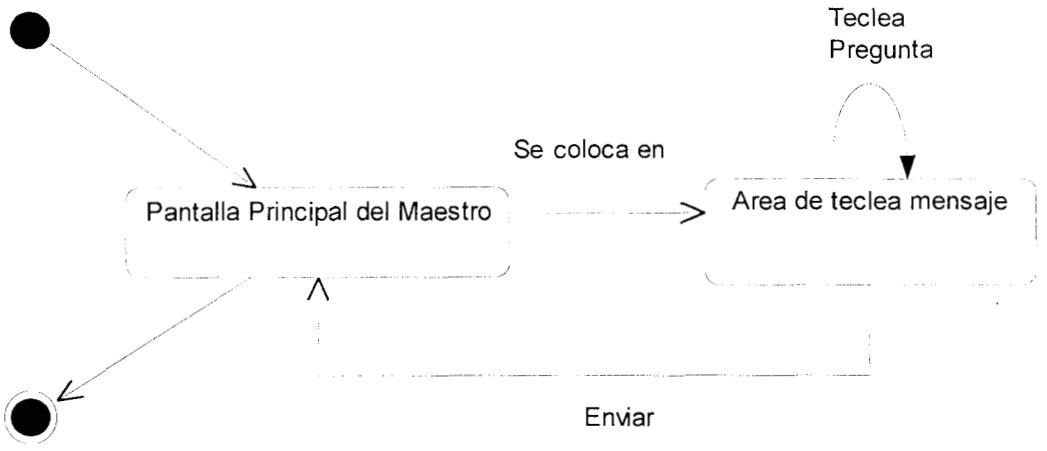
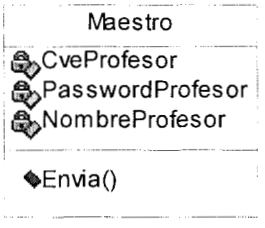


Diagrama de Navegación de Pantallas.



Modelo del Dominio del Problema.

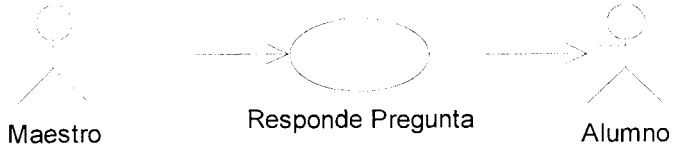
Paso	Descripción UC Maestro Pregunta
1	**Maestro** *teclea* **pregunta** en el **área de teclea mensaje a enviar.**
2	**Maestro** *presiona* enviar.
3	**Sistema** *envia* la **pregunta** a todos los **alumnos.**
4	Fin del caso.



5.2.2.2 Modelo de Requerimientos: Use Case Maestro Responde Pregunta

Modelo de Casos

El siguiente diagrama de casos se muestra como el maestro puede responder a la pregunta de un alumno.



Descripción del caso:

Paso	Descripción UC Maestro Responde Pregunta
1	Sistema trae al área de mensajes la pregunta del Alumno.
2	Maestro teclea la respuesta dentro del área de teclea mensaje a enviar.
3	Maestro presiona enviar.
4	Sistema envia a todos los alumnos la respuesta.
5	Fin del caso.

Modelo de Interfaz.

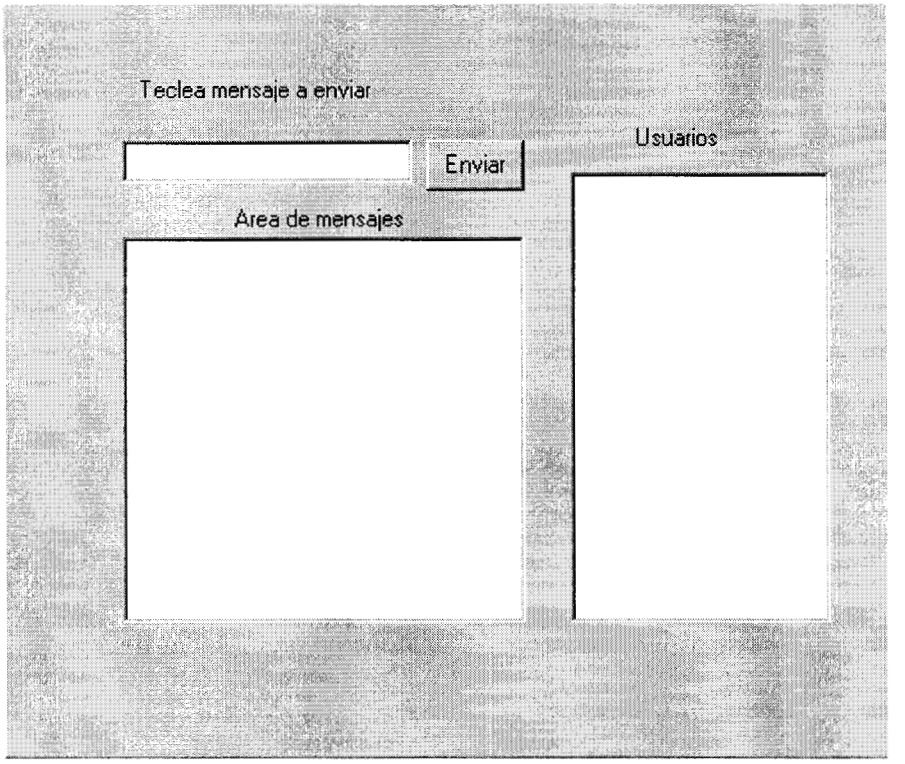
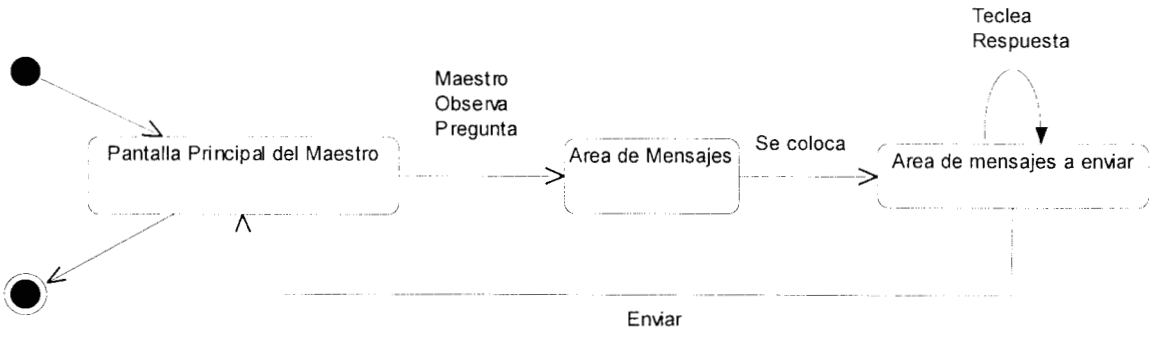
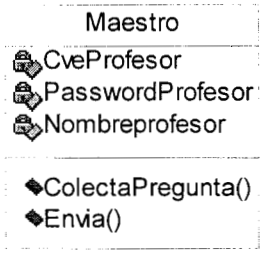


Diagrama de Navegación de Pantallas



Modelo del Dominio del Problema.

Paso	Descripción UC Maestro Responde Pregunta
1	**Sistema** *trae* al **área de mensajes** la **pregunta** del **Alumno**.
2	**Maestro** *teclea* la **respuesta** dentro del **área de teclea mensaje a enviar**.
3	**Maestro** *presiona* enviar.
4	**Sistema** *envia* a todos los **alumnos** la **respuesta.**
5	Fin del caso.



5.2.3. Modelo de Requerimientos: Use Case ValidaAlumno

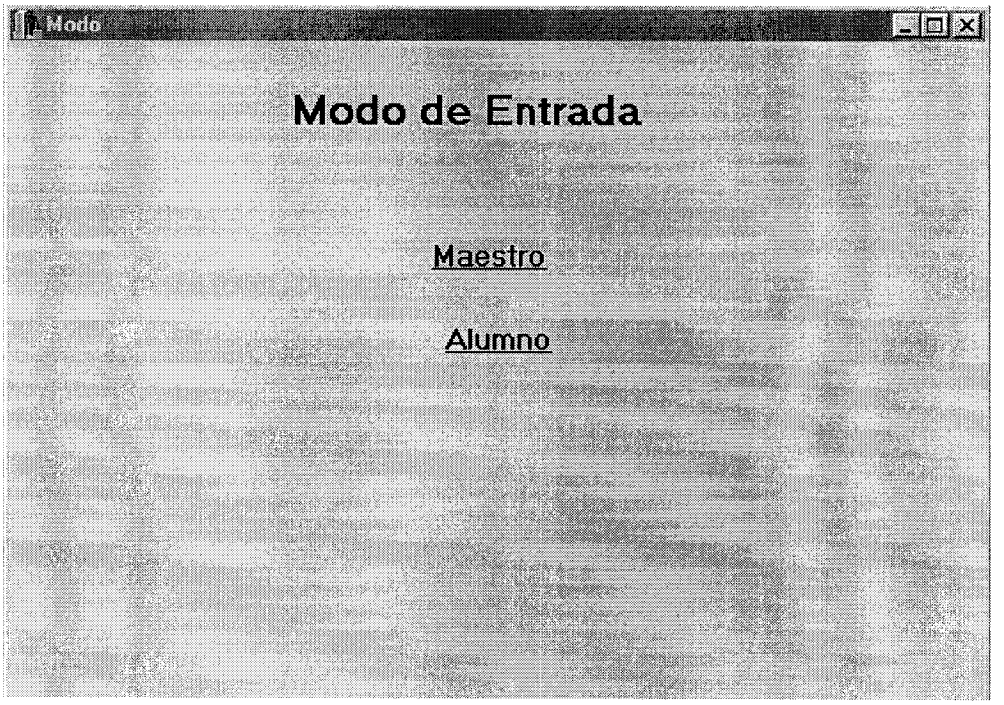
Modelo de Casos.

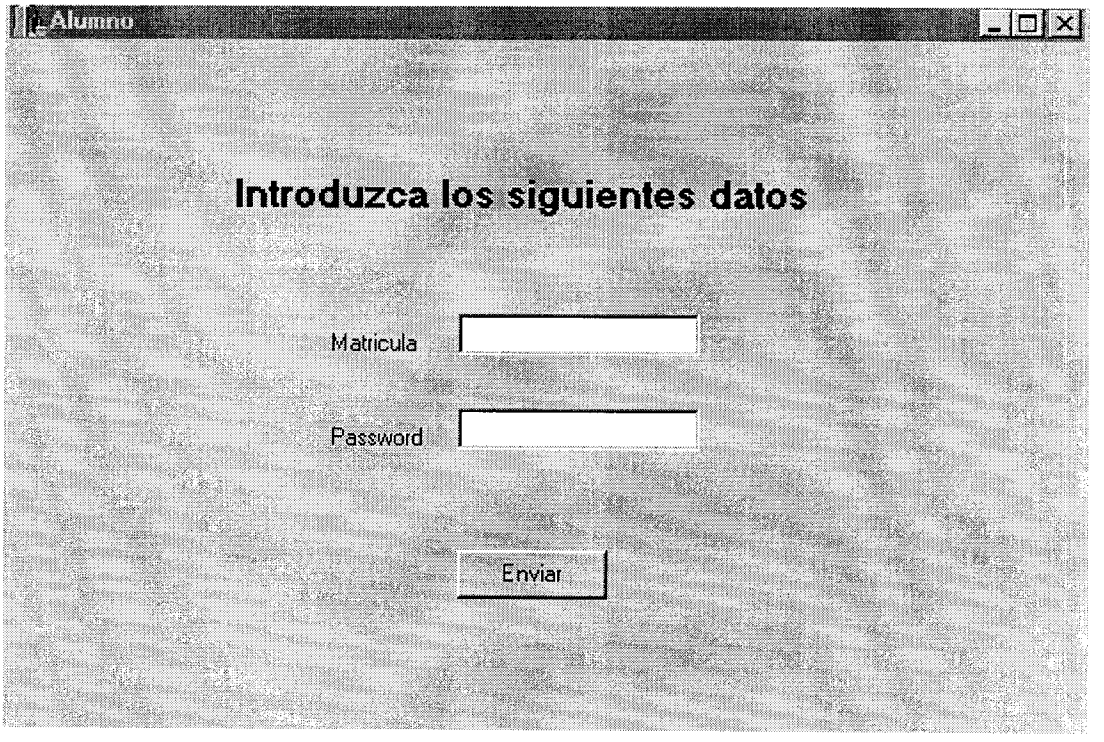
El siguiente diagrama de casos muestra la validación que tendrá que realizar el alumno, cada vez que ingrese al sistema.

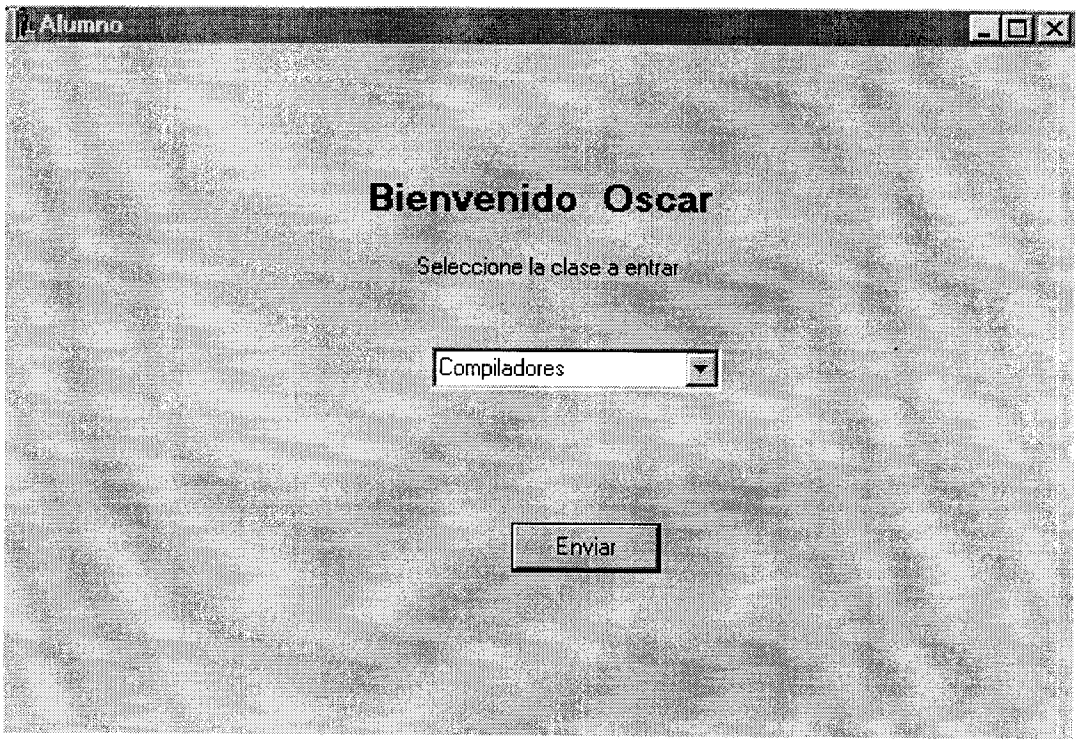
Descripción del caso:

Paso	Descripción UC Valida Alumno
1	El sistema despliega una lista de modo de entrada.
2	El alumno selecciona el modo de entrada alumno.
3	El sistema despliega una pantalla donde el alumno introduce su matricula, password y presiona enviar.
4	El sistema valida la matricula y el password.
5	Si los datos son correctos el sistema despliega una lista de cursos donde el alumno esta dado de alta, para tomar clases.
6	El alumno elige el curso al que desea ingresar y presiona enviar.
7	El alumno entra al Aula Virtual.
8	Fin del Caso.

Modelo de Interfaz.






Alumno
Bienvenido Oscar
Seleccione la clase a entrar
Compiladores
Enviar

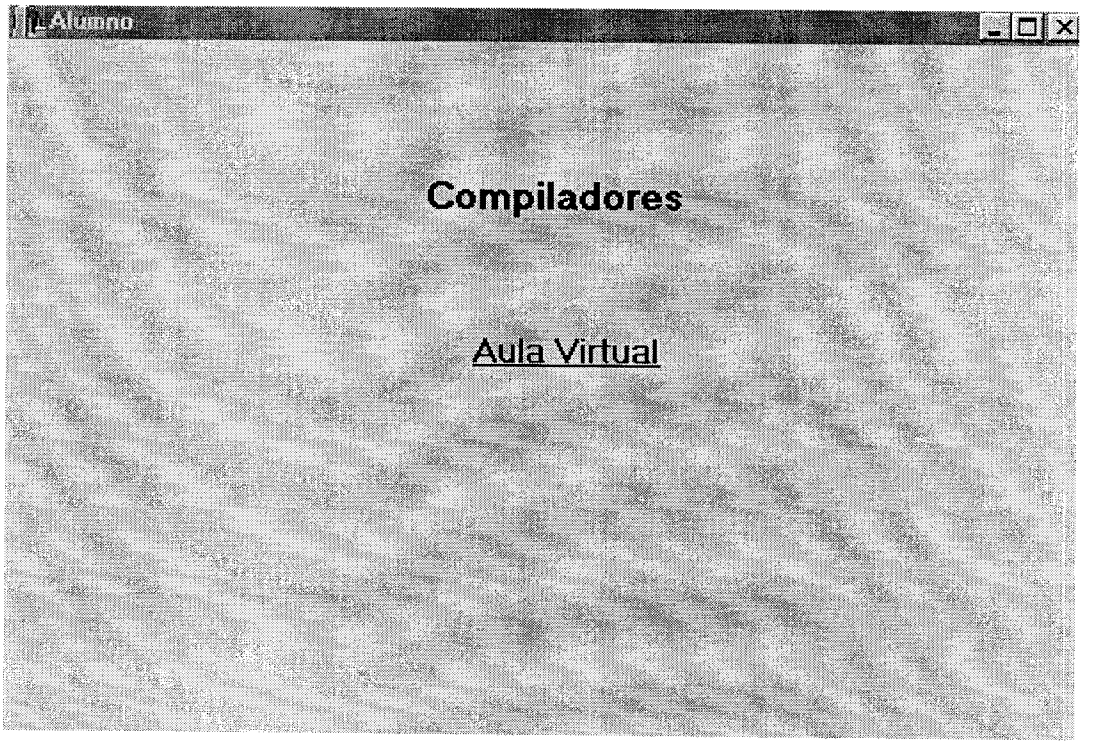
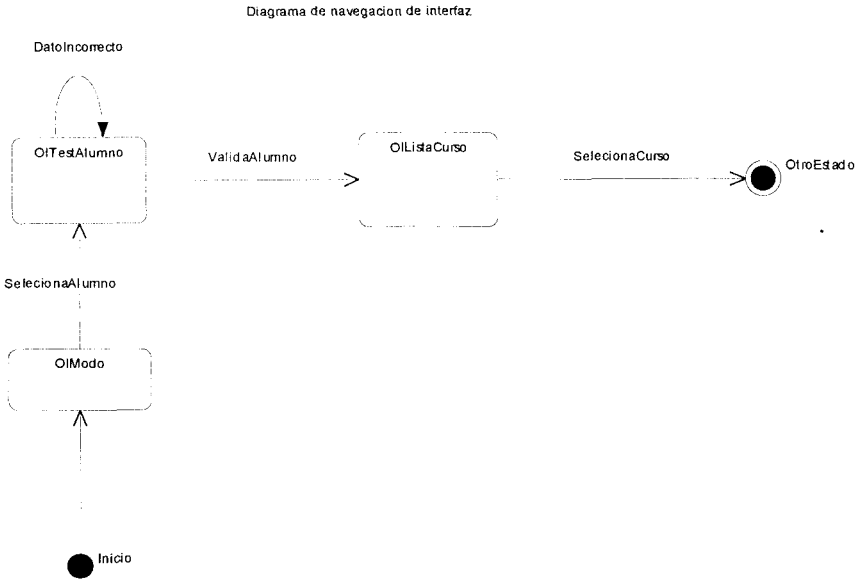
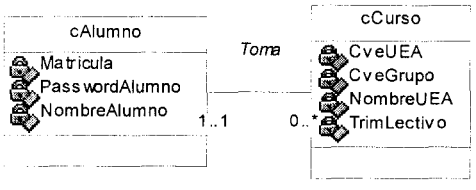

Alumno
Compiladores
Aula Virtual

Diagrama de Navegación de Pantallas



Modelo del Dominio del Problema.

Paso	Descripción UC Valida Alumno
1	El **sistema** *despliega* una **lista** de **modo de entrada**.
2	El **alumno** *selecciona* el **modo de entrada** alumno.
3	El **sistema** *despliega* una **pantalla** donde el **alumno** *introduce* su **matricula, password** y *presiona enviar.*
4	El **sistema** *valida* la **matricula** y el **password.**
5	Si los **datos** son correctos el **sistema** *despliega* una **lista de cursos** donde el **alumno** esta dado de alta, para *tomar clases.*
6	El **alumno** *elige* el **curso** al que *desea ingresar* y *presiona enviar.*
7	El **alumno** entra al **Aula Virtual.**
8	Fin del Caso.



5.2.4 Modelo de Requerimientos: Use Case AlumnoTomaClase

Este Use Case tiene dos especializaciones, que son: \

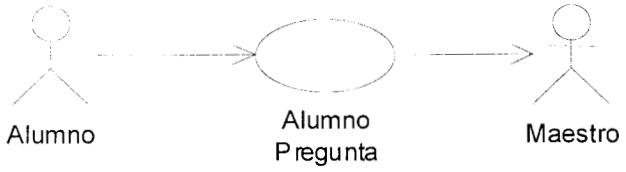
- Use Case Alumno Pregunta
- Use Case Alumno Responde Pregunta

Estos Uses Cases se describen a continuación.

5.2.4.1 Modelo de Requerimientos: Use Case Alumno Pregunta

Modelo de Casos.

El siguiente diagrama de casos muestra como el alumno puede realizar una pregunta al maestro.



Descripción del caso:

Paso	Descripción UC Alumno Pregunta
1	Alumno teclea pregunta en el area de teclea mensaje.
2	Alumno presiona enviar.
3	Sistema lleva pregunta al maestro y a los demás alumnos.
4	Fin del caso.

Modelo de Interfaz.

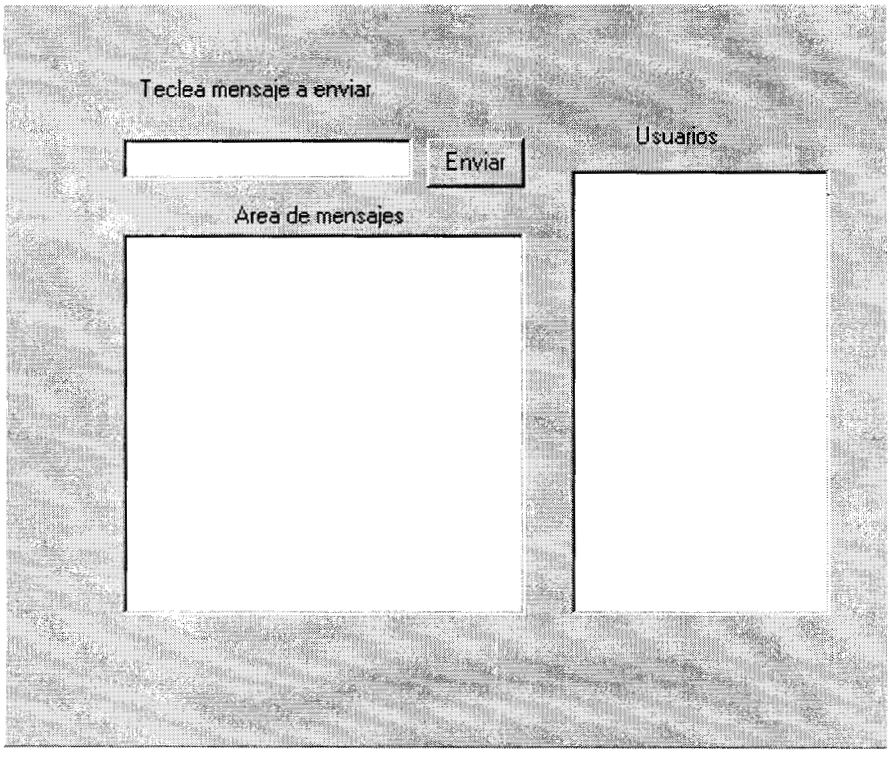
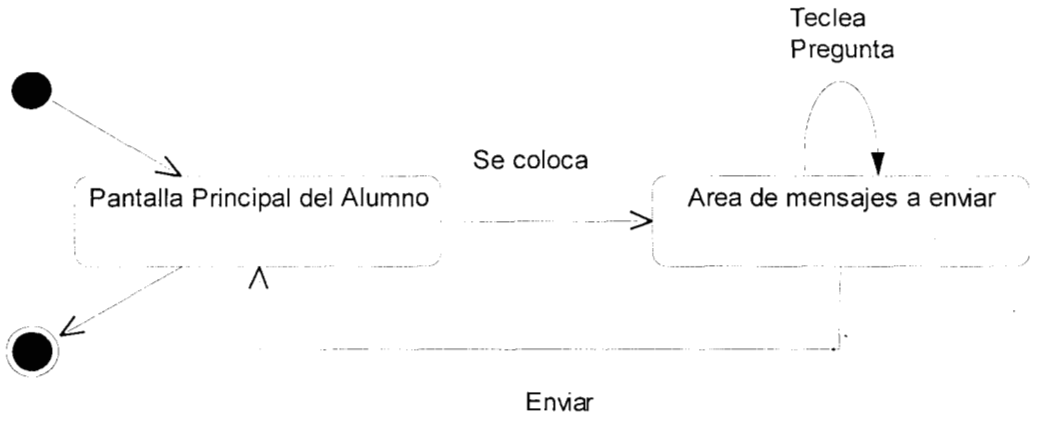
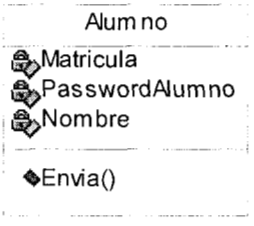


Diagrama de Navegación de Pantallas



Modelo del Dominio del Problema.

Paso	Descripción UC Alumno Pregunta
1	**Alumno** *teclea* pregunta en el **area de teclea mensaje a enviar**.
2	**Alumno** *presiona* enviar.
3	**Sistema** *lleva* pregunta al **maestro** y a los demás **alumnos**.
4	Fin del caso.



5.2.4.2 Modelo de Requerimientos: Use Case Alumno Responde Pregunta

Modelo de Casos.

El siguiente diagrama de casos muestra como un alumno puede contestar la pregunta realizada por el maestro.

Descripción del caso:

Paso	Descripción UC Alumno Responde Pregunta
1	Sistema trae al área de mensajes la pregunta del Maestro.
2	Teclea la respuesta dentro del área de teclea mensaje.
3	El Alumno presiona enviar.
4	El Sistema envia al maestro la respuesta junto con el nombre del alumno.
5	Fin del caso.

Modelo de Interfaz.

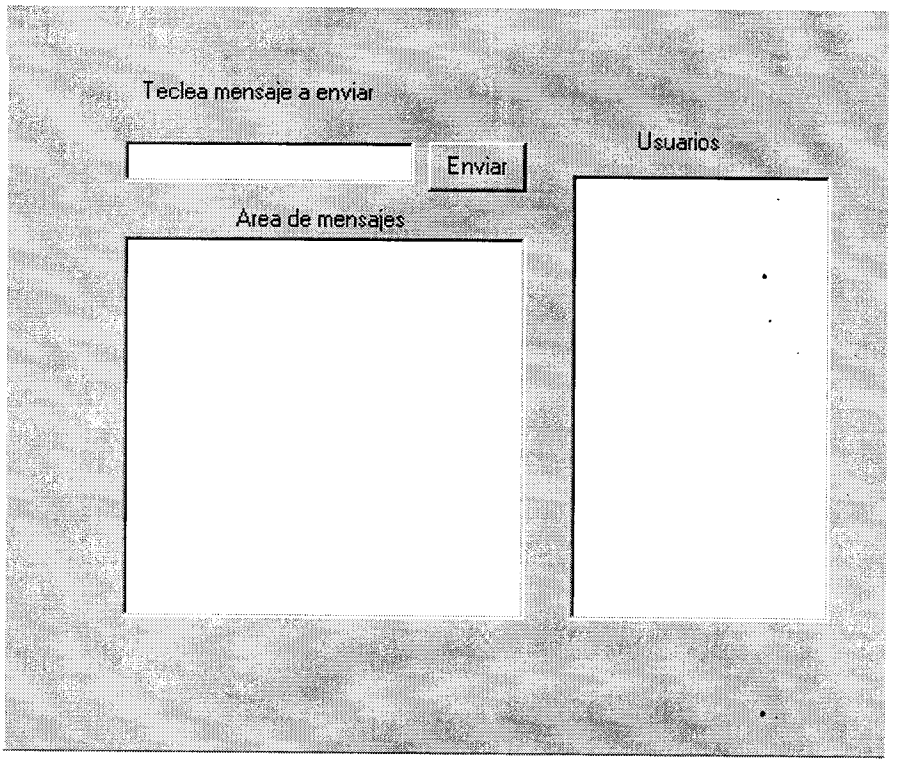
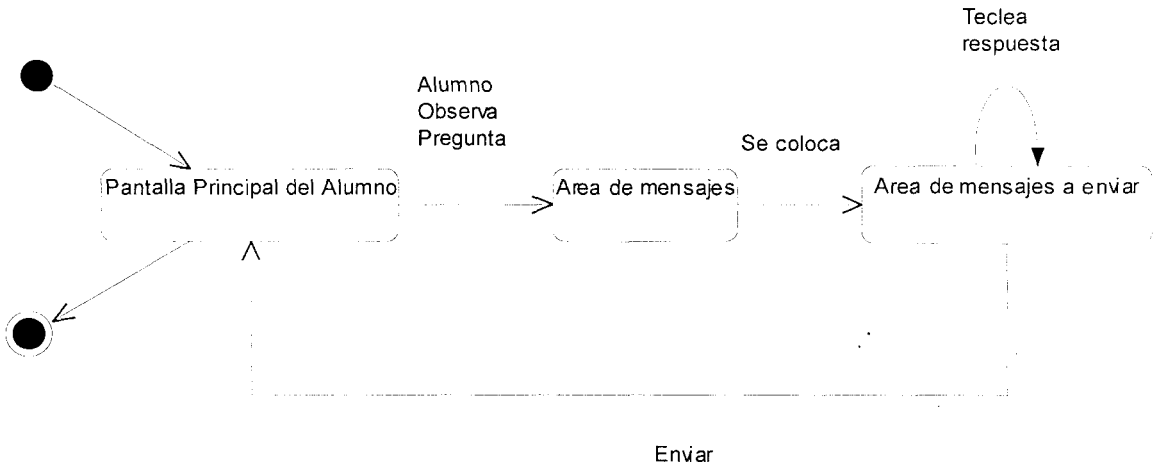
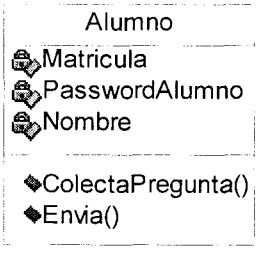


Diagrama de Navegación de Pantallas.



Modelo del Dominio del Problema.

Paso	Descripción UC Alumno Responde Pregunta
1	**Sistema** trae al **área de mensajes** la **pregunta** del **Maestro**.
2	El **Alumno** *teclea* la **respuesta** dentro del **área de teclea mensaje de enviar**.
3	El **Alumno** *presiona* enviar.
4	El **Sistema** *envia* al **maestro** la **respuesta** junto con el **nombre del alumno.**
5	Fin del caso.



5.3 Modelo de Análisis para cada Use Case.

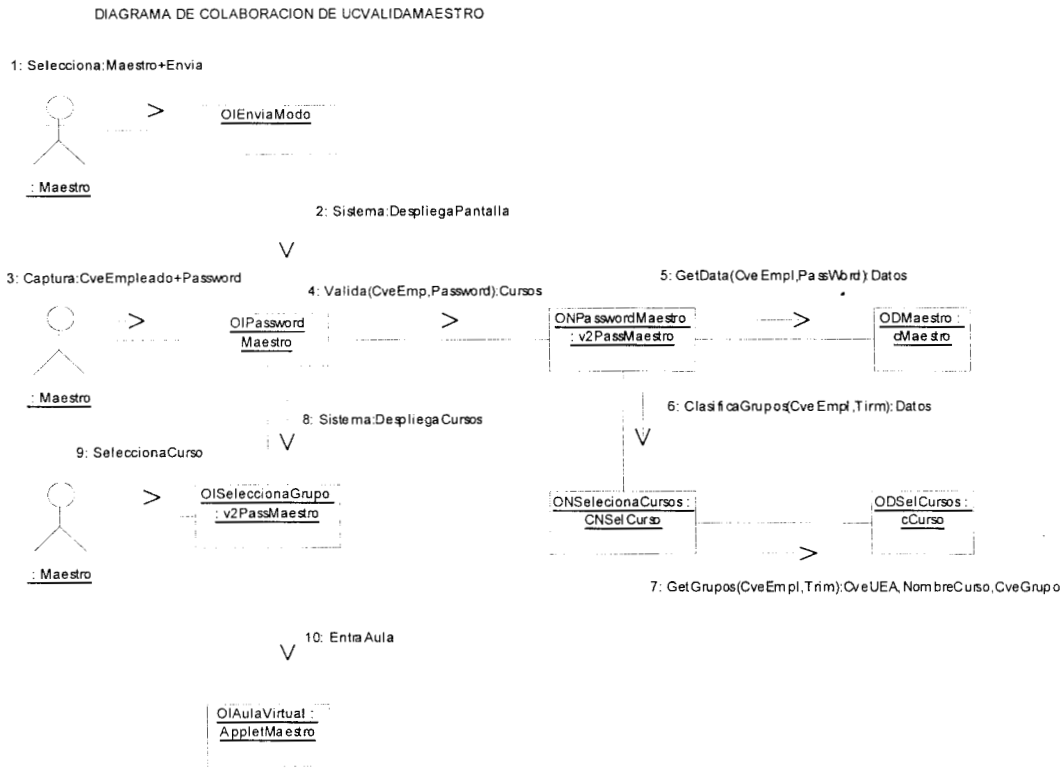
5.3.1 Escenarios explorados.

Los escenarios que se exploraron, corresponden únicamente al encontrado en el marco de "Simple Correcto", ya que al ingresar a otro escenario, en el que la

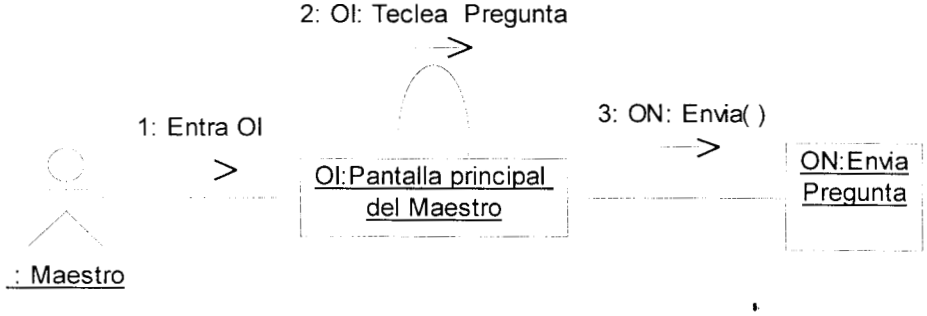
clave de acceso del alumno o el profesor sean incorrectas, el sistema simplemente no entra en ejecución, por lo que se omitieron diagramas que no tiene gran valor ilustrativo.

Diagramas de Colaboración.

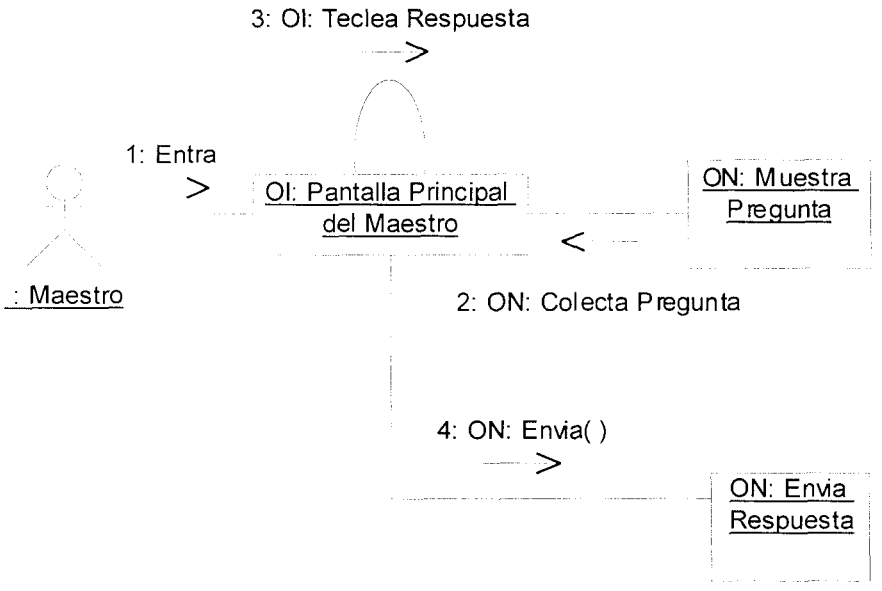
Modelo de Análisis: Use Case ValidaMaestro



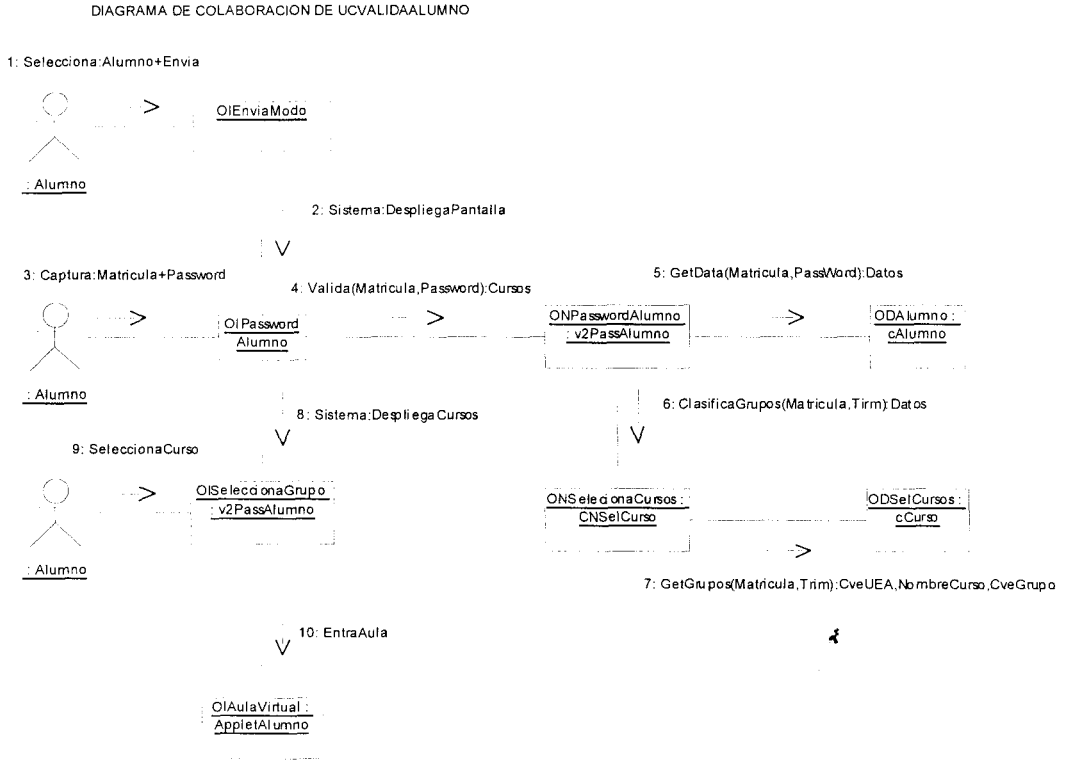
Modelo de Análisis: Use Case MaestroPregunta



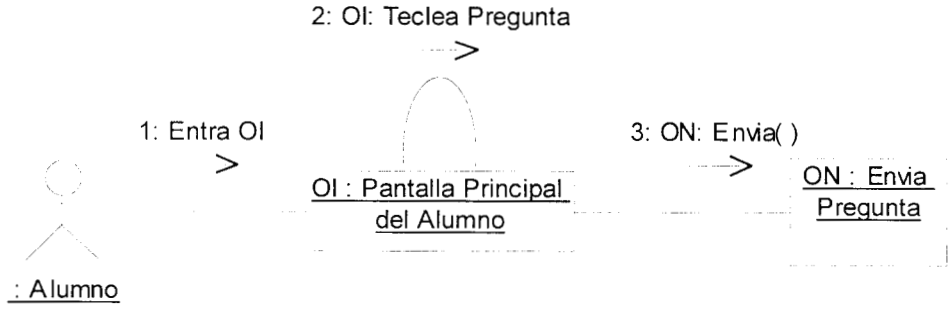
Modelo de Análisis: Use Case MaestroRespondePregunta



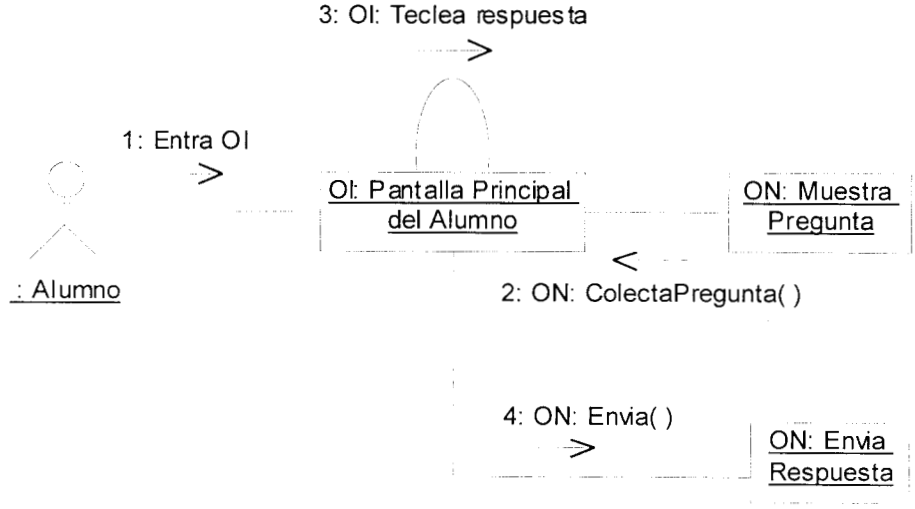
Modelo de Análisis: Use Case ValidaAlumno



Modelo de Análisis: Use Case Alumno Pregunta

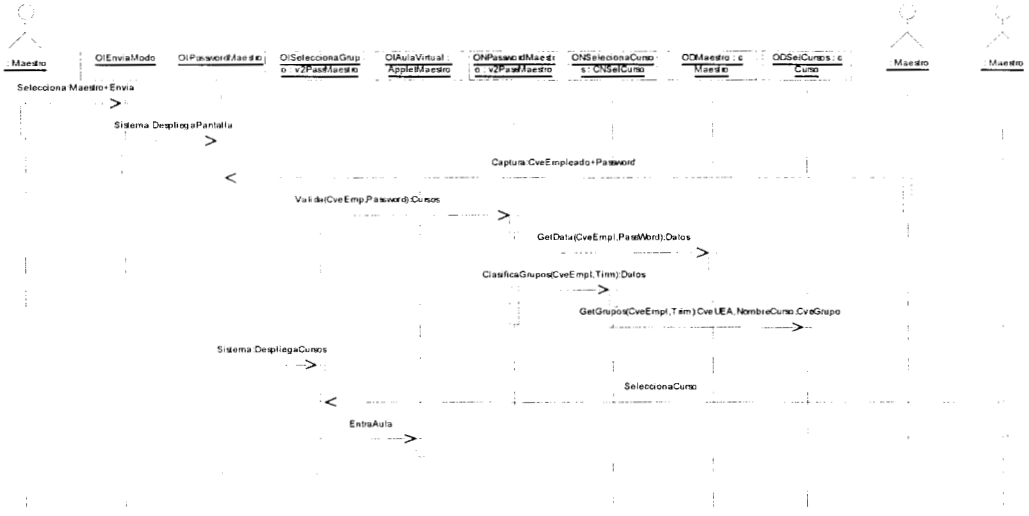


Modelo de Análisis: Use Case Alumno Responde Pregunta

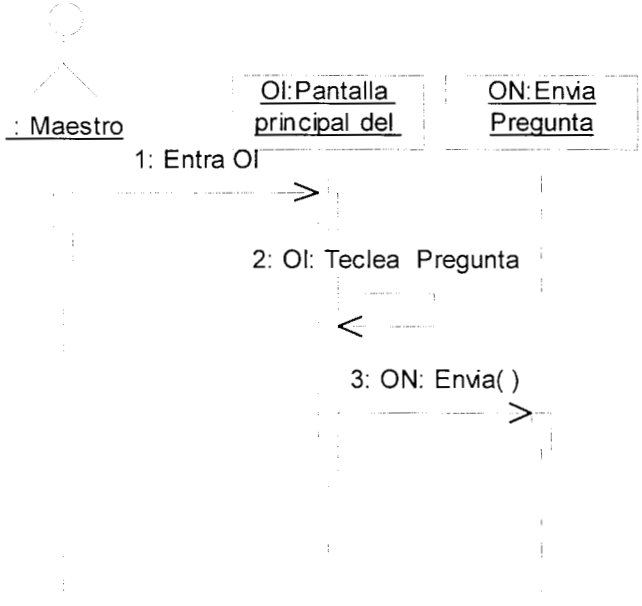


5.4. Modelo de Diseño.

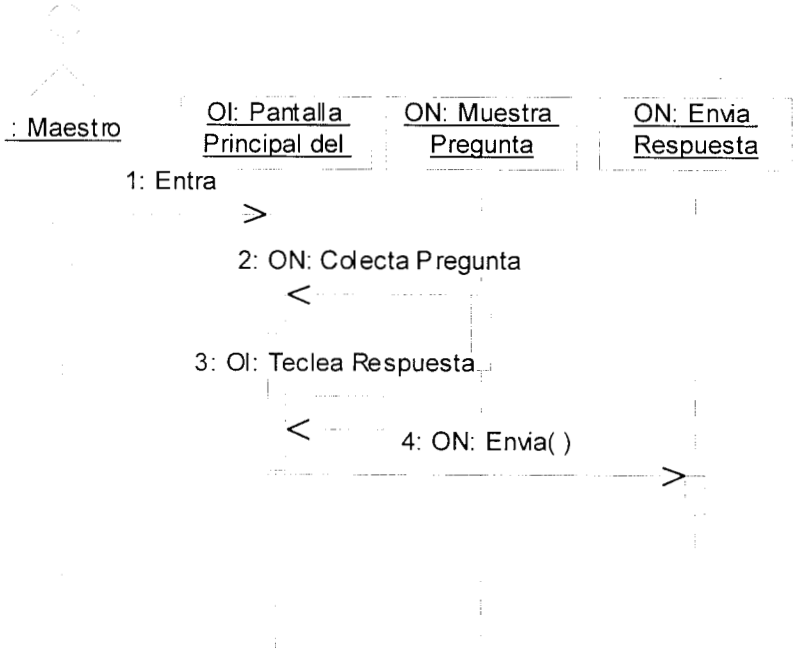
Modelo de Diseño: Use Case ValidaMaestro.



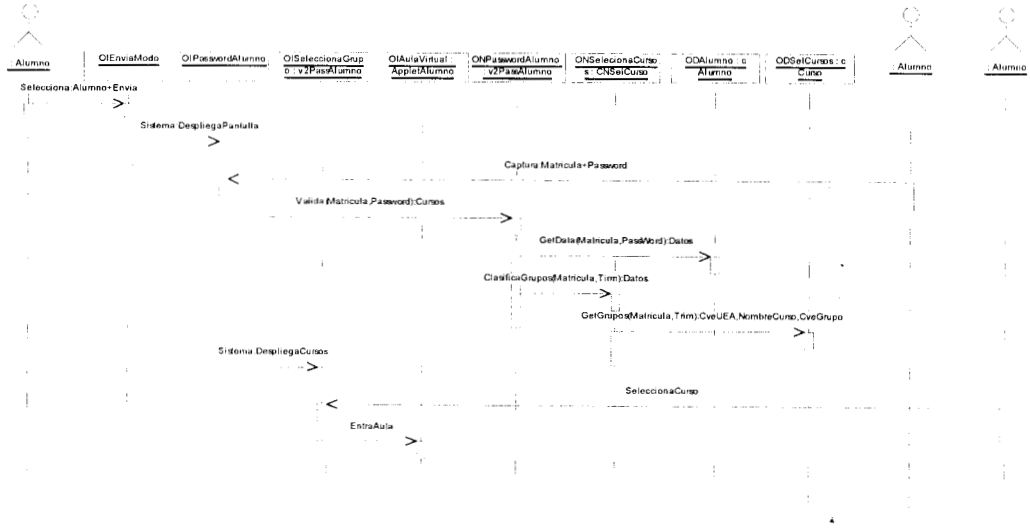
Modelo de Diseño: Use Case Maestro Pregunta.



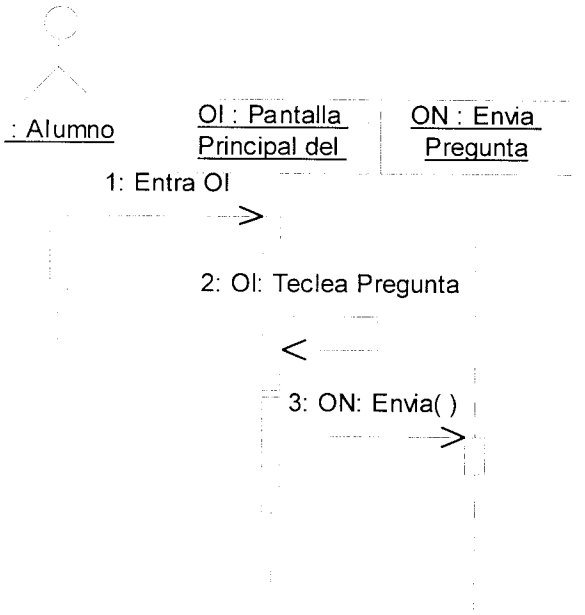
Modelo de Diseño: Use Case Maestro Responde Pregunta.



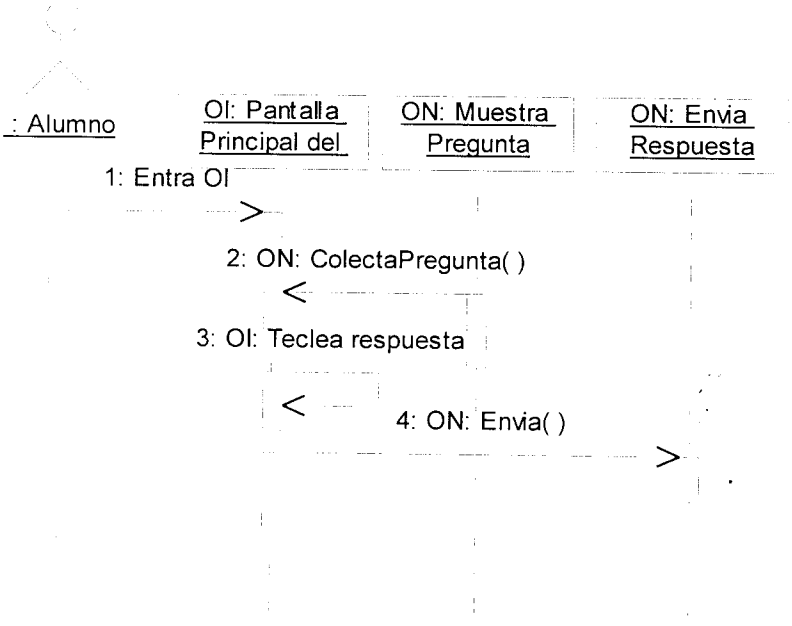
Modelo de Diseño: Use Case ValidaAlumno.



Modelo de Diseño: Use Case Alumno Pregunta.



Modelo de Diseño: Use Case Alumno Responde Pregunta.



5.4.1 Modelo de Implantanción.

La codificación de todo el análisis anterior se realizo en Java, con la ayuda de Servlets, y Applets, además es importante mencionar que también se requirieron paginas HTML para la presentación en la red del proyecto Educación a Distancia.

6. Conclusiones.

El proyecto de Educación es una necesidad de muchas Universidades, pero no solo a nivel nacional, si no a nivel mundial, esta necesidad a provocado el desarrollo de este proyecto que habia llegado a un primer prototipo, sin embargo en este trimestre se ha llegado a un segundo prototipo en el cual se logro una comunicación completa via internet, es decir se logró el intercambio de información mediante el uso de la clase virtual entre el alumno y el maestro.

Es importante mencionar que algunos factores obstruyeron la elaboración de dicho proyecto, como lo fueron : la falta de información, la inestabilidad del servidor e incluso la inestabilidad de los requisitos.

De la realización del presente proyecto, se pudo observar y tener viva conciencia de la importancia y el impacto en la sociedad de la denominada "Educación a Distancia". Asumiéntose a la Universidad Virtual como una aplicación, en esta era de las comunicaciones, ya que es una forma flexible de enseñanza especializada a distancia que utiliza avanzadas tecnologias, esto permite ofrecer una formación de excelencia, cumpliendo con los estandares academicos internacionales.

Entre otras cosas hay que destacar que este proyecto brinda comodidad a los alumnos, ya que pueden tomarse las clases en cualquier parte donde exista Internet, por lo que puede ser muy comercial.

7. Bibliografía.

- 1001 tips para programar en Java, Steven W. Griffith et al. Mc Graw Hill , México ,D.F ,1998, 1ra. Edición Español.

- Java Servlets, Kalr Moss, Mc Graw Hill, United States of America, 1999, first edition (Primera Edición).

- Aprendiendo Borland Jbuilder2 en 21 días, Donald Doherty y Michelle M. Manning, Prentice may Hispanoamericana S.A., M{exico, 2000, 1ra. Edición Español.

REFERENCIAS

- http://developer.java.sun.com/developer/onlineTraining/Servlets/Fundamentals/contents.html
- http://java.sun.com/docs/books/tutorial/index.html
- http://www.netverk.com.ar/usuarios/suweb/html/manualde/man000.htm
- http://members.es.tripod.de/froufe/parte2/cap2-1.html
- http://java.sun.com/products/jdk/1.2/docs/api/
- http://geminis.adi.uam.es/~dcc/html/html1.htm
- http://www.eis.uva.es:80/GuiaHTML/introHTML.html