Comunidades virtuales en un entorno de programación libre
- Calendario de Congresos -

Marcos Patricio Amate
Consultor: Julià Minguillón Alfonso
Curso 2004/2005 (junio)

Universitat Oberta de Catalunya
Ingeniería Técnica en Informática de Gestión

A Marta por su amor, y por creer plenamente en mí.

A Víctor que es la alegría de todos nosotros.

A toda mi familia, sin ella no hubiese tenido el tiempo suficiente para poder sacar la carrera adelante.

También no puedo olvidarme de Juanmi y especialmente de Salva compañeros de fatigas en las verdes y en las maduras.

A todos ellos mil gracias.

Índice

1.- Introducción

1.1- Finalidad del Trabajo Final de Carrera

En este trabajo final de carrera se pretende que el alumno trabaje en un proyecto desde su inicio hasta su final, pasando por todas sus fases de desarrollo. En este desarrollo he trabajado en un proyecto relacionado con el diseño, programación e implementación de herramientas para la generación de comunidades virtuales en Internet y para la administración de contenidos dinámicos, utilizando la separación por capas que propone la arquitectura J2EE.

Concretamente en este proyecto, he generado una herramienta para el mantenimiento de un calendario de congresos, lo que permitirá a los profesores de la Universita Oberta de Catalunya poder gestionar los congresos a los cuales deberán asistir, de una manera clara y sencilla en un entorno web.

Para poder realizar el proyecto es necesario que se tuviese conocimientos de JAVA, de análisis y diseño orientado a objetos, por lo que era recomendable que se hubieran cursado las asignaturas de programación de la carrera. Además era recomendable tener conocimientos básicos de la arquitectura multicapa J2EE, de bases de datos relacionales así como de lenguaje HTML.

1.2.- Objetivos del Trabajo Final de Carrera

1.2.1.- Objetivo general

Análisis, diseño e implementación de un calendario que permita gestionar el mantenimiento de eventos relacionados con unos congresos utilizando una arquitectura de 3 capas basada en J2EE.

Se pretende separar claramente la presentación, la lógica de negocio y el acceso a datos. Concretamente, la presentación se prevé implementar con combinación de servlets y JSPs, la lógica de negocio y la capa de datos con Java Beans.

1.2.2.- Alcance del proyecto

Funcionalidad general de un calendario, teniendo en cuenta la limitación de tiempo impuesta en su desarrollo. Se considera prioritaria la calidad del diseño frente a los detalles de implementación o la estética de la interfaz de usuario.

1.2.3.- Metodología

El desarrollo del proyecto seguirá de manera simplificada el ciclo de vida del Rational Unified Process, RUP, con sus etapas: Inicio, elaboración, construcción y transición. En cada etapa habrá una recogida de requisitos, análisis y diseño, realización y una prueba.

En la fase de análisis el proceso se dirige por la identificación de las clases de entidad, sus atributos y la identificación de las relaciones entre clases, realizando los casos de uso. En la fase de diseño se eliminan las herencias entre clases, se diseña la persistencia y la interfaz gráfica de usuario. En la fase de implementación se ponen en práctica la tecnología J2EE y las páginas JSP.

1.2.4.- Producto final. Resultados esperados.

Al final del proyecto se obtiene una aplicación J2EE que nos permitirá gestionar mediante un calendario todo lo referente al mundo de los congresos, fechas clave, acontecimientos, preparación de viaje, hoteles, asistencias, etc.

Con el uso de este lenguaje, la UOC da la posibilidad al estudiante de la Ingeniería Técnica de Informática de Gestión, de desarrollar una herramienta para que unos investigadores puedan saber en todo momento que fechas se acercan y si fuese necesario enviar algo, como preparar el viaje, etc. En definitiva en el desarrollo de un calendario que gestione toda la planificación utilizando, para ello, la plataforma J2EE.

Utilizando esta tecnología se busca una solución integral que responda a las necesidades actuales y futuras de los investigadores, diseñándola de tal forma que interaccionen de forma transparente o, bien, que se dejen bases sólidas para la implementación futura de soluciones complementarias.

En este marco, la plataforma de este proyecto informático se diseña para un funcionamiento multicapa, donde además del manejador de bases de datos y teniendo como capas intermedias un servidor de http y Web Caché, se definen múltiples capas de aplicación desarrolladas con el estándar J2EE.

Otras ventajas con que cuenta son la escalabilidad, la posibilidad de tener la lógica del negocio independiente con los clientes, la alta disponibilidad de las aplicaciones y del acceso a las bases de datos, así como el manejo de pequeños clientes y del factor seguridad, entre otros.

Entre los principales beneficios que se pueden desencadenar con la aplicación de J2EE, encontramos:

- Enriquecimiento y robustez de soluciones.
- Generación de componentes reutilizables dentro del ámbito de los distintos sistemas del proyecto, con lo cual se puede reducir hasta un 50% los tiempos de desarrollo proyectado.
- Detección de mecanismos de seguimiento integrales al proyecto.
- Disminución de tiempos de investigación y de elaboración de prototipos.
- Reducción de tiempos en la replicación del conocimiento.

El proyecto se ha realizado siguiendo la tecnología J2EE, Plataforma JavaTM 2, Edición Empresarial (J2EE) que define el estándar para el desarrollo de aplicaciones de múltiples capas.

J2EE simplifica las aplicaciones empresariales basándolos sobre componentes estandartizados, modulares, proporcionando un juego completo de servicios a aquellos componentes, y maneja muchos detalles de comportamiento de aplicación automáticamente, sin programación compleja.

1.3.- Planificación del Trabajo Final de Carrera

1.3.1.- Recursos

Los recursos humanos se limitan a una persona, con una dedicación de unas 100h, sin contar formación al cliente destinatario final de la aplicación.

Requerimientos de software:

- J2SDK 1.4.1

- JBoss 3.0.7 incluyendo su servidor web Tomcat Apache.

- mySQL como sistema gestor de base de datos.

- IntelliJ-Idea 3.0.4 como herramienta de desarrollo en Java.

1.3.2.- Riesgos del proyecto

Se consideran los riesgos derivados de la inexperiencia en implementación de aplicaciones J2EE. Concretamente la inexperiencia puede conducir a diseños incorrectos, ineficaces o difíciles de implementar, así como a ralentizar la implementación. Para mitigar estos riesgos será necesario desde el principio del proyecto dedicar un tiempo adicional a la formación, mediante tutoriales y libros

online. También se espera compensar la inexperiencia con el seguimiento de soluciones propuestas por otros, en forma de patrones.

1.3.3.- Calendario

El calendario irá marcado por las fechas de las entregas:

- 8 de marzo: plan del proyecto.
- 12 de abril: documento de análisis.
- 17 de mayo: documento del diseño.
- 20 de junio: entrega final, consistente en una memoria, una presentación en diapositivas y el producto final, la aplicación web.

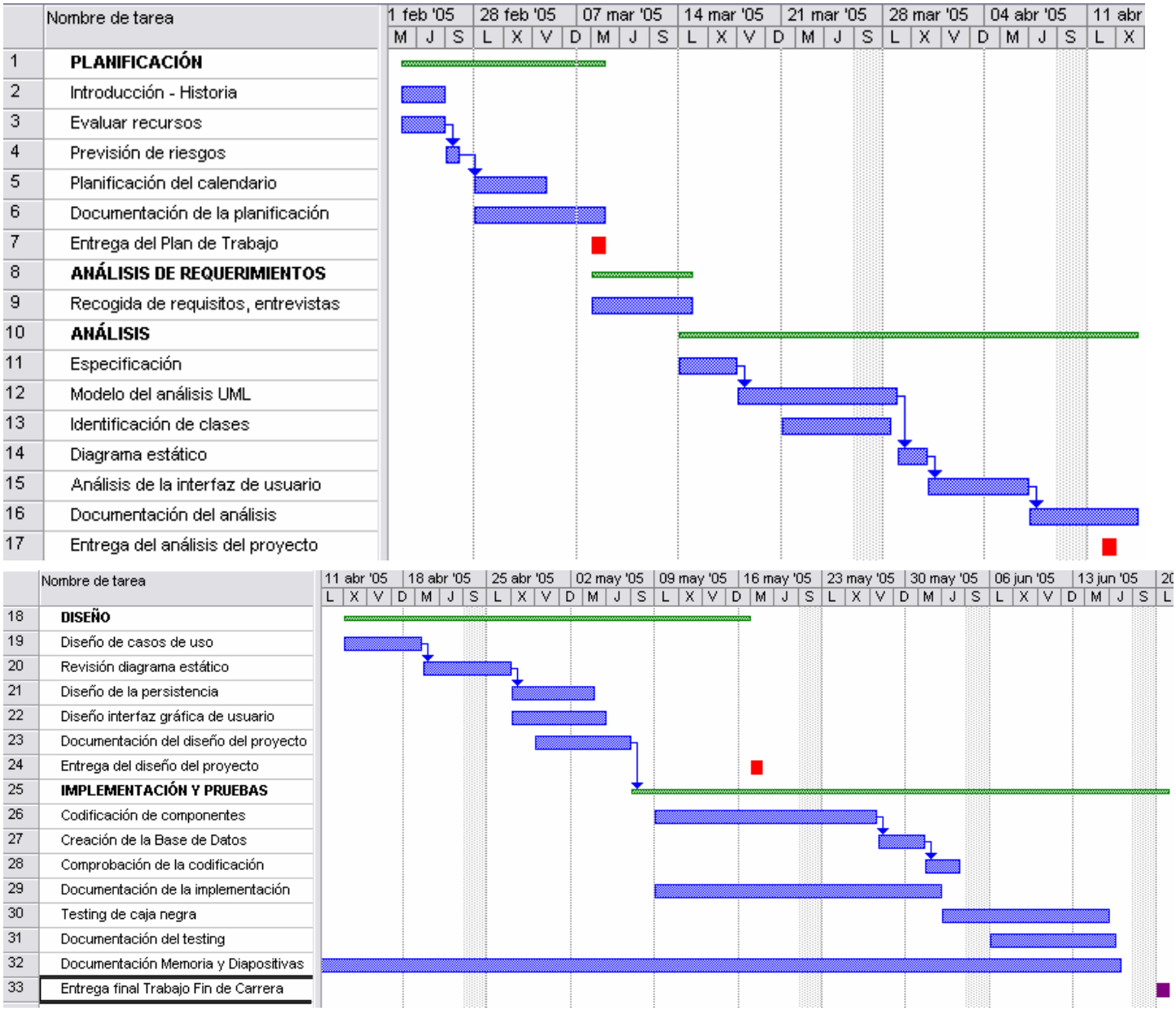
El calendario de las fases y tareas se detalla seguidamente en un diagrama de Gantt.

1.3.4.- Fases del proyecto

El desarrollo del proyecto seguirá aproximadamente el método RUP, debidamente simplificado considerando el contexto de este proyecto como un TFC. Las fases del proyecto serán

Planificación, Análisis, Diseño, Implementación y Pruebas. Los tiempos estimados para cada tarea y su temporización es la que se representa en los diagramas de Gantt de la página siguiente.

Figura 1: diagramas de Gantt.



1.3.5.- Tabla de trabajos por semanas

Semana	Fechas	Trabajos
1	23/02/05 – 27/02/05	Introducción – Historia Evaluar recursos Previsión de riesgos Documentación memoria final y diapositivas
2	28/02/05 – 06/03/05	Planificación del calendario Documentación de la planificación Documentación memoria final y diapositivas
3	07/03/05 – 13/03/05	Documentación de la planificación Entrega PAC 1 Plan de Trabajo (08/03/05) Recogida de requisitos, entrevistas Documentación memoria final y diapositivas
4	14/03/05 – 20/03/05	Recogida de requisitos, entrevistas Especificación Modelo de análisis UML Documentación memoria final y diapositivas
5	21/03/05 – 27/03/05	Modelo de análisis UML Identificación de clases Documentación memoria final y diapositivas
6	28/03/05 – 03/04/05	Identificación de clases Diagrama estático Análisis de la interfaz de usuario Documentación memoria final y diapositivas
7	04/04/05 – 10/04/05	Análisis de la interfaz de usuario Documentación del análisis Documentación memoria final y diapositivas
8	11/04/05 – 17/04/05	Documentación del análisis Entrega PAC 2 Análisis del proyecto (12/04/05) Diseño de casos de uso Documentación memoria final y diapositivas
9	18/04/05 – 24/04/05	Diseño de casos de uso Revisión diagrama estático Documentación memoria final y diapositivas

10	**25/04/05 – 01/05/05**	**Revisión diagrama estático** **Diseño de la persistencia** **Diseño de la interfaz gráfica de usuario** **Documentación del diseño del proyecto** **Documentación memoria final y diapositivas**
11	**02/05/05 – 08/05/05**	**Diseño de la persistencia** **Diseño de la interfaz gráfica de usuario** **Documentación del diseño del proyecto** **Documentación memoria final y diapositivas**
12	**09/05/05 – 15/05/05**	**Codificación de componentes** **Documentación de la implementación** **Documentación memoria final y diapositivas**
13	**16/05/05 – 22/05/05**	**Codificación de componentes** **Entrega PAC 3 Diseño del proyecto (17/05/05)** **Documentación de la implementación** **Documentación memoria final y diapositivas**
14	**23/05/05 – 29/05/05**	**Codificación de componentes** **Documentación de la implementación** **Creación de la base de datos** **Documentación memoria final y diapositivas**
15	**30/05/05 – 05/06/05**	**Documentación de la implementación** **Creación de la base de datos** **Comprobación de la codificación** **Testing de caja negra** **Documentación memoria final y diapositivas**
16	**06/06/05 – 12/06/05**	**Testing de caja negra** **Documentación del testing** **Documentación memoria final y diapositivas**
17	**13/06/05 – 19/06/05**	**Testing de caja negra** **Documentación del testing** **Documentación memoria final y diapositivas**
18	**20/06/05**	**Entrega Final: TRABAJO FIN DE CARRERA**

1.4.- Breve historia de J2EE

Muy buenas historias comienzan así y la del J2EE no es la excepción, ya que nos transporta al año de 1995, cuando la compañía norteamericana Sun Microsystems® introduce al mercado la primera plataforma de software universal diseñada desde y para el crecimiento de Internet y de las Intranets corporativas. Esta tecnología fue denominada Java y permite a los programadores escribir aplicaciones una vez, así como hacerlas correr en cualquier ordenador lo que, desde entonces, ha revolucionando al mundo del desarrollo de software por representar un cambio de paradigma.

Pero ¿cómo está esto del cambio de paradigma? Con Java las cosas cambian, en tanto es una plataforma de software, que independiente del hardware, se aloja en el sistema mientras exista el elemento conocido como "Máquina Virtual de Java" (MVJ, o JVM por sus siglas en inglés). La MVJ hace posible que una aplicación desarrollada en Java se ejecute en cualquier sistema operativo que soporte esta máquina virtual, como por ejemplo Unix, Linux, Macintosh®, Windows, entre otros.

La principal misión de la MVJ es garantizar la portabilidad de las aplicaciones Java y especificar las instrucciones, llamadas bytecodes, que se pueden ejecutar. El intérprete Java entonces ejecuta las órdenes que se guardan en los archivos cuya extensión es .class.

1.4.1.- Beneficios

Java, como lenguaje de programación, reúne todas las características que debiera tener un ambiente orientado a objetos: es sencillo, cuenta con capacidad de generación de aplicaciones distribuidas, es robusto, seguro, de arquitectura neutral, portátil, multi-hilo, dinámico y de alto rendimiento.

Los programas más comunes que se pueden hacer con Java son, por una parte, los applets, que son pequeños programas de aplicación que se ejecutan en el navegador de la máquina cliente; y aplicaciones, esto es, programas ejecutados directamente en la MVJ.

Pero esto no es todo, la API (Application Program Interface) de Java es muy rica, ya que está formada por un conjunto de paquetes de clases que proporcionan una gran funcionalidad. Adicionalmente, Java también incluye extensiones que, por ejemplo, permiten definir un API para tercera dimensión, manejo de los servidores, telefonía y reconocimiento de voz, entre otras cuestiones.

El núcleo de la API viene con cada una de las implementaciones de la MVJ: tipos de datos, clases y objetos, applets, manejo de la red (networking), seguridad, y componentes (Java Beans). Este último concepto —los componentes— es básico para

la comprensión del tema que voy a desarrollar en este Trabajo Final de Carrera: J2EE.

1.4.2.- Nace J2EE: Java 2 Enterprise Edition

Debido a la necesidad del mercado de desarrollo de software, respecto a contar con medios y herramientas que permitan construir aplicaciones “empresariales”, se diseñó la plataforma abierta y estándar de Java, mejor conocida como Java 2 Enterprise Edition, (J2EE). Se le denomina plataforma porque proporciona especificaciones técnicas que describen el lenguaje pero, además, provee las herramientas para implementar productos de software (aplicaciones) basados en dichas especificaciones.

J2EE ha sido diseñada para aplicaciones distribuidas que son construidas con base en componentes (unidades funcionales de software), los cuales interaccionan entre sí para formar parte de una aplicación J2EE. Un componente de esta plataforma debe formar parte de una aplicación y ser desplegado en un contenedor, o sea en la parte del servidor J2EE que le ofrece al componente ciertos servicios de bajo nivel y de sistema (tales como seguridad, manejo de concurrencia, persistencia y transacciones). J2EE no es sólo una tecnología, sino un estándar de desarrollo, construcción y despliegue de aplicaciones.

La plataforma J2EE resulta una propuesta atractiva, interesante y de vanguardia que responde, de manera natural, a la demanda actual para el desarrollo de software, bajo el concepto de arquitectura en capas.

Para una organización que cuenta con una diversidad de equipos de cómputo, con sus respectivos sistemas operativos, una diversidad de “ambientes de trabajo” en las distintas áreas y una diversidad de “entornos de desarrollo” (los sistemas y soluciones informáticas con que cuentan en cada área), la propuesta de J2EE para “unificar” el desarrollo de aplicaciones que puedan utilizarse en la empresa, resulta viable y con posibilidades de amplia aceptación, sobre todo por el incentivo principal de Java: Write Once, Run Anywhere - Programa una vez, ejecuta en cualquier equipo.

1.4.3.- Retos

Portabilidad. Los componentes deben poder ser “reubicados” del entorno operativo o, bien, del entorno de trabajo (middleware), e incluso del sistema, sin requerir cambios significativos.

Diversidad de ambientes. La empresa actual cuenta con más de un tipo de infraestructura de hardware/software que debe aprovecharse de manera conjunta en aplicaciones de uso común o empresarial.

Oportunidad en su aparición. Los componentes que se desarrollan deben ser implementados y publicados para su integración a las aplicaciones, en el momento en

que son requeridos. Si no contamos con el componente, podemos estar afectando el proceso productivo de la organización.

Reutilización. El diseño de componentes de software, con independencia de los servicios que proveen al “exterior”, permite reforzar el concepto de reutilización, que no significa solamente “cortar y pegar” partes del código, sino que proporciona al integrador de aplicaciones la facilidad de generar soluciones completas de software, aprovechando los beneficios de los componentes disponibles.

2.- Recogida de requisitos, entrevistas

2.1.- Introducción

La recogida de requisitos busca obtener información sobre dos aspectos:

1. Los procesos que se tienen que hacer encima de los datos
2. La manera como se tiene que pedir a los usuarios los datos de entrada y que función de la aplicación quieren utilizar en cada momento y presentarles los resultados.

2.2.- Los requisitos

Los requisitos son la especificación de lo que tiene que hacer la aplicación, son descripciones del comportamiento, propiedades y restricciones de la aplicación que son necesarias desarrollar.

Tienen un doble papel sirven de base para un acuerdo entre los usuarios (cliente) y los desarrolladores del software y también son la información de salida para poder desarrollar el programa.

2.3-.- Fuentes de información

La principal fuente de información para poder desarrollar un programa, son las entrevistas y encuestas a los futuros usuarios. Una buena documentación sobre el sistema actual, si ya estuviera informatizado el sistema, y conocer otros puntos de vista, por ejemplo preguntar a colegas de los usuarios. Otra fuente de información podría ser realizar una revisión de sistemas parecidos que haya en el mercado.

En este proyecto, la principal fuente de información serán las preguntas a mi Consultor Julià, el cual mediante el uso del correo electrónico vía bidireccional, me irá facilitando toda la información requerida por mí.

2.4.- El contexto de la aplicación

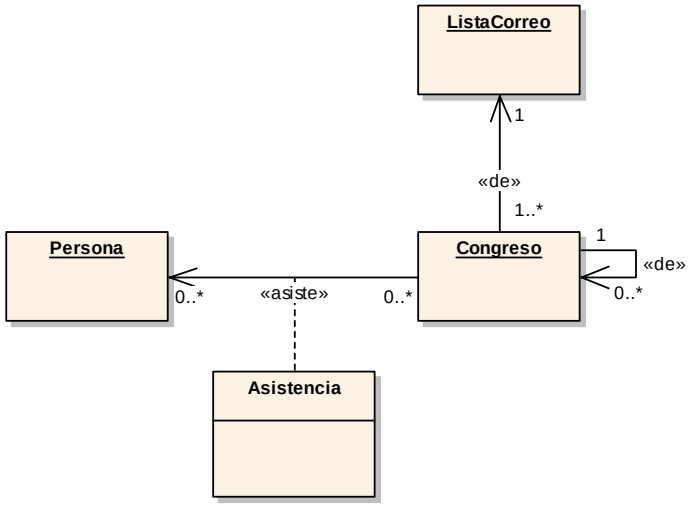
Los desarrolladores de software, normalmente no conocen la actividad profesional de los usuarios. Y si es así sería necesario que adquiriesen rápidamente un cierto conocimiento desde el punto de vista organizativo, es el contexto de la aplicación.

2.5.- El modelo del dominio

El modelo del dominio recoge los tipos de objetos, clases, más importantes. Por una parte tenemos los objetos de negocio (facturas, expedientes, cuentas, etc.), por otra los objetos del mundo real (cliente, historial de ventas, etc.), y por último tenemos acontecimientos (llegada de trenes, término de plazos, etc.)

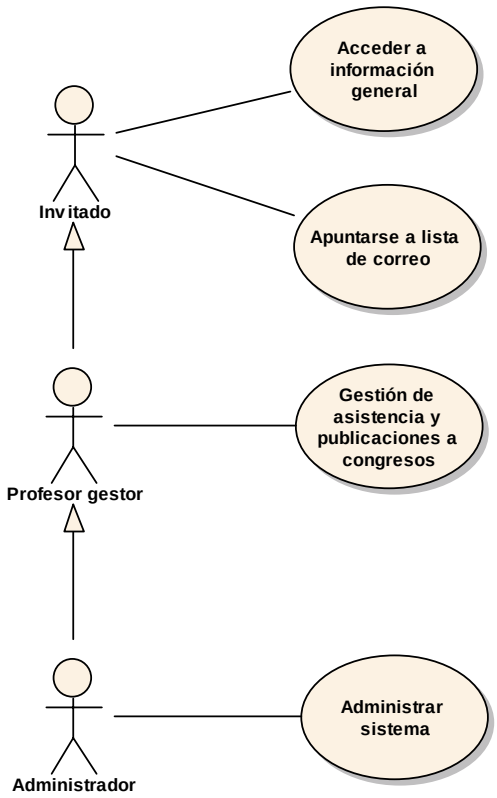
El modelo de dominio del Trabajo Final de Carrera, para la aplicación de Congresos, viene determinado por:
Los objetos o clases que a primera vista veo son: 'persona', 'congreso', 'listacorreo', 'asistencia'.

El modelo estático del modelo de dominio sería:



2.6.- El modelo de negocio

Este modelo describe a grandes rasgos los procesos y entidades principales en el entorno de la aplicación, en términos de casos de uso y unidades de trabajo que intervienen. El diagrama de casos de uso del modelo de negocio



2.7.- Glosario del modelo de negocio

Administrador: Usuario privilegiado que tendrá un perfil que le permitirá administrar todo el sistema.
Usuario: Usuario final de la aplicación, puede ser Administrador (privilegios de todo), Profesor gestor (usuarios registrados, acceso a parte privada), o Invitado (cualquiera, acceso a parte pública).
Congreso: Acontecimiento que tiene lugar para debatir unos conocimientos
Lista de Correo: Lista en donde pueden apuntarse los usuarios para que sean informados de los acontecimientos de los congresos.
Asistencia: Acción de un Usuario Registrado (Profesor gestor) que confirma la asistencia a un determinado congreso.
Publicación: Aportaciones que los Profesores gestores aportan para dar una ponencia.

2.8.- Los guiones

Son las operaciones más frecuentes que los usuarios realizan en su trabajo. Son sesiones que un actor lleva a cabo con relación a la aplicación. Puede haber muchos guiones posibles y sirven para poder sacar los casos de uso.

Guión Administrador*:* mantenimiento de usuarios, verificación de errores, mantenimiento del sistema, ver información de un congreso, ver la lista de correos, ver el control de asistencias a congresos, ver las publicaciones de un congreso.

Guión Profesor Gestor: ver información general de los congresos, apuntarse a lista de correos, acceso a gestión de congresos y publicaciones, apuntarse a un congreso, darse de baja de un congreso, enviar información de un congreso.

Guión Invitado: ver información general de los congresos, apuntarse a lista de correos.

2.9.- Identificación de los actores

Un actor es un papel de cualquier entidad externa que se prevé que interactuará con la aplicación y le dará información o recibirá. A una persona pueden corresponderle diversos actores. Se supone que los actores no llevan a cabo ninguna secuencia de casos de uso determinada. Cada actor se tiene que corresponder al menos con un usuario concreto. No es necesario que se detallen los actores, basta con identificarlos. No tiene sentido que dos actores intervengan exactamente en los mismos casos de uso.

Tipos de actores:

1. Usuario final
2. Usuario privilegiado o gestor del sistema
3. Entorno informático

En nuestro caso tenemos tres actores:

Administrador, Profesor gestor e Invitado.

Todos ellos son usuarios finales

2.10.- Identificación de los casos de uso

Los casos de uso son:

- Procesos autónomos iniciados por un actor o por otro caso de uso.
- Representan funciones ofrecidas por la aplicación e identifican las entradas y las salidas.
- Describen el qué de estas funciones y no el cómo.
- Pueden servir de base para pruebas de caja negra.
- Los casos de uso que se describan por primera vez en una iteración determinada tienen que encajar con los de las iteraciones anteriores.

En nuestro caso identifico los siguientes casos de uso:

1. Consultar congreso
2. Apuntarse a lista de correo

3. Darse de baja de la lista de correo
4. Apuntarse a un congreso
5. Darse de baja de un congreso
6. Enviar documentación
7. Alta congresos
8. Baja de congresos
9. Modificación de congresos
10. Alta de usuarios
11. Baja de usuarios
12. Modificación de usuarios
13. Consulta de usuarios
14. login

2.11.- Identificación de las relaciones entre casos de uso

Relaciones de extensión: ligadas a una condición, cuando hay diversos flujos de procesos posibles o bien casos especiales o errores que tienen que ser tratados de manera diferente.

En nuestro caso tenemos por una parte:
Dos relaciones de extensión, los casos de uso Enviar documentación y Darse de baja de un congreso, solo los realizarán si se han apuntado a un congreso, de ahí la relación de extensión.
Dos relaciones de extensión entre Darse de baja de la lista de correo y Apuntarse a la lista de correo, ya que si no te has dado de alta no podrás darte de baja. Y otra entre Consultar congreso y Dar de alta congreso.
Dos relaciones de extensión entre Alta de congresos y los casos de uso Baja de congresos y Modificación de Congresos, ya que si no has dado de alta el congreso no podrás dar de baja o modificar un congreso.
Y por último tres relaciones de extensión entre Alta de usuarios y los casos de uso Modificación de usuarios, Baja de usuarios y Consulta de usuarios, ya que si no has dado de alta el usuario difícilmente, podrás hacer estas últimas operaciones.

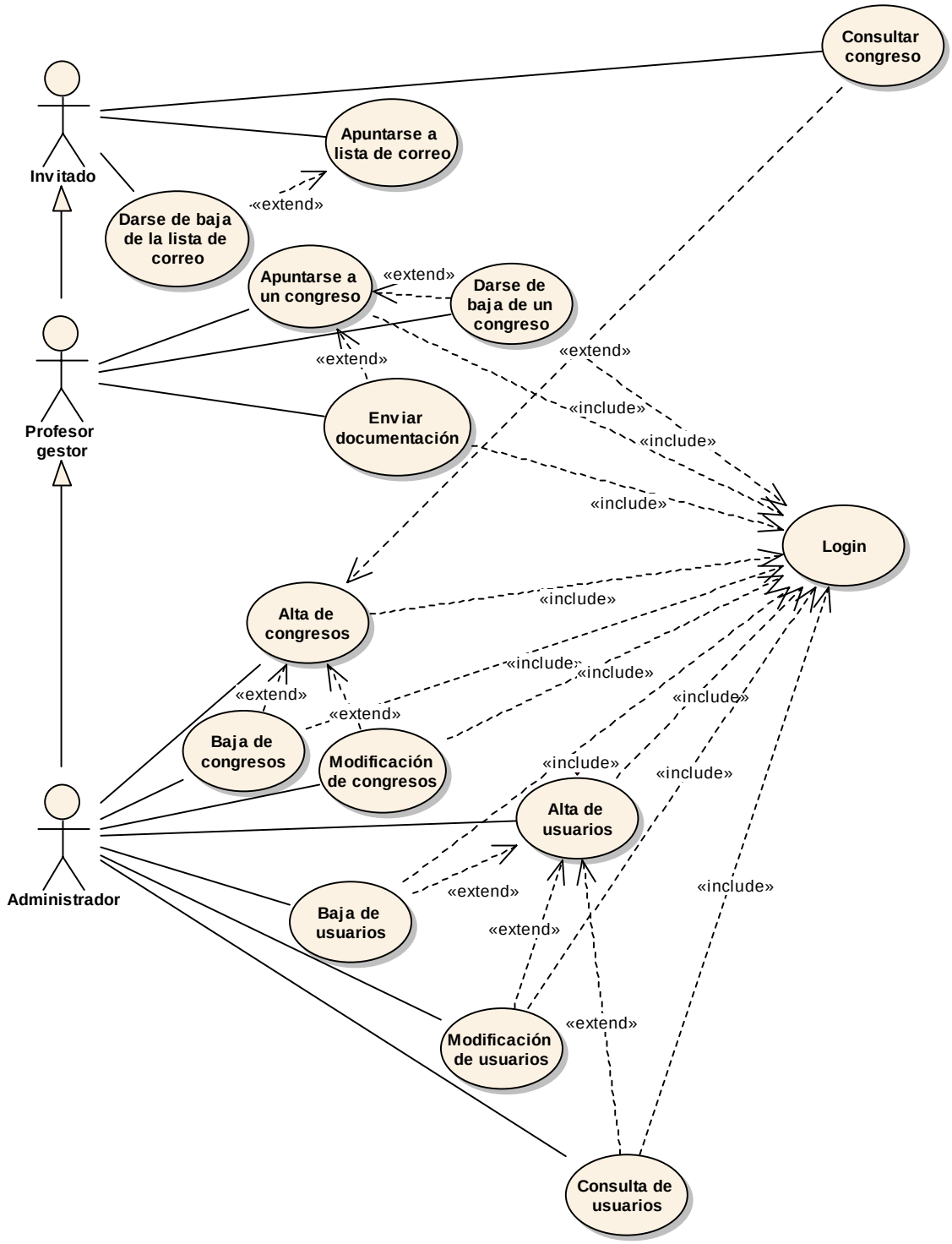
Relaciones de inclusión: es un recurso para evitar tener que escribir una misma parte del proceso dentro de diversos casos de uso.

En nuestro caso nos encontramos con las relaciones de inclusión derivadas del proceso de login, para que los profesores gestores, y los administradores puedan efectuar sus operaciones una vez que se hayan identificado en el sistema.

Relaciones de especialización: de los dos casos de uso relacionados uno es una versión especializada del otro.

A continuación en la página siguiente obtenemos el diagrama de Casos de Uso.

2.12.- Diagrama de Casos de uso:



2.13.- Documentación textual de los casos de uso

2.13.1.- Caso de uso número 1: "Consultar congreso"

Resumen de la funcionalidad: Busca el congreso que el usuario ha seleccionado mediante el calendario, o mediante la introducción de palabras claves en el formulario pertinente.
Papel dentro del trabajo del usuario: es un caso de uso principal en el *invitado*, y ocasional en el *Profesor gestor* y en el *Administrador.*
Actores: Administrador, Profesor gestor e Invitado
Casos de uso relacionados: Alta congreso
Precondición: ninguna
Poscondición: ninguna, es una consulta
Busca el *congreso* que cumple con la condición especificada en el calendario o en el formulario, y se obtiene una ficha pública del congreso (si ha accedido por Calendario) o un listado de los Congresos con la palabra clave si ha accedido por menú.

2.13.2.- Caso de uso número 2: "Apuntarse a lista de correo"

Resumen de la funcionalidad: cada usuario selecciona sus congresos y da una dirección de correo, a la cual le enviarán información detallada de esos congresos.
Papel dentro del trabajo del usuario: es un caso de uso ocasional tanto de *invitado*, *Profesor gestor* y *Administrador.*
Actores: Administrador, Profesor gestor e Invitado
Casos de uso relacionados: Darse de baja de la lista de correo
Precondición: ninguna
Poscondición: el usuario ha quedado inscrito en la lista de correo de congresos
El usuario seleccionará los congresos y una vez que los tenga seleccionados el sistema le pedirá que introduzca su correo electrónico para poderle enviar información detallada de los congresos.

2.13.3.- Caso de uso número 3: "Darse de baja de la lista de correo"

Resumen de la funcionalidad: El usuario se da de baja de la lista de correo para no recibir más información de los congresos que había seleccionado.
Papel dentro del trabajo del usuario: es un caso de uso ocasional tanto de *invitado*, *Profesor gestor* y *Administrador.*
Actores: Administrador, Profesor gestor e Invitado
Casos de uso relacionados: Apuntarse a lista de correo.
Precondición: Que el usuario esté apuntado a la lista de correo
Poscondición: el usuario se ha dado de baja de la lista de correo de congresos
El usuario seleccionará la opción de darse de baja de la lista de congresos, el sistema le pedirá que introduzca su correo electrónico para poderle dar de baja.

2.13.4.- Caso de uso número 4: "Apuntarse a un congreso"

Resumen de la funcionalidad: Un usuario registrado y autentificado se apunta a un congreso para asistir de oyente o de ponente.
Papel dentro del trabajo del usuario: es el caso de uso principal del actor *Profesor gestor*, y ocasional del actor *Administrador*.
Actores: Administrador, Profesor gestor.
Casos de uso relacionados: Planificar viaje, Enviar documentación, Login y Darse de baja de un congreso
Precondición: el congreso tiene que existir y el usuario no tiene que estar ya inscrito a ese congreso, y además tiene que estar registrado e identificado.
Poscondición: el usuario queda apuntado para ese congreso determinado.
El usuario busca el congreso al cual quiere apuntarse, y si está autentificado en el sistema, podrá rellenar el formulario con sus datos para poder apuntarse al congreso.

2.13.5.- Caso de uso número 5: "Darse de baja de un congreso"

Resumen de la funcionalidad: Un usuario registrado y autentificado se da de baja de un congreso apuntado con anterioridad.
Papel dentro del trabajo del usuario: es el caso de uso principal del actor *Profesor gestor*, y ocasional del actor *Administrador*.
Actores: Administrador, Profesor gestor.
Casos de uso relacionados: Planificar viaje, Enviar documentación, Login y Baja de un congreso
Precondición: el congreso tiene que existir y el usuario tiene que estar apuntado a ese congreso, además el usuario tiene que estar registrado e identificado.
Poscondición: el usuario queda desligado a ese congreso determinado.
El usuario busca el congreso al cual quiere borrarse, y si está autentificado en el sistema, una vez seleccionado se borrará del mismo.

2.13.6.- Caso de uso número 6: "Enviar documentación"

Resumen de la funcionalidad: A través de aquí el usuario podrá enviar sus publicaciones si las tiene previstas de llevar al congreso.
Papel dentro del trabajo del usuario: es un caso de uso ocasional del actor *Profesor gestor*, y del actor *Administrador*.
Actores: Administrador, Profesor gestor.
Casos de uso relacionados: Apuntarse a un congreso y Login
Precondición: el congreso tiene que existir y el usuario tiene que estar apuntado a ese congreso y estar registrado e identificado.
Poscondición: el usuario ha enviado publicaciones al congreso.
El usuario busca el congreso al cual quiere enviar publicaciones, y si está autentificado en el sistema y apuntado a ese congreso, podrá rellenar el formulario con sus datos para poder enviar documentación al congreso.

2.13.7.- Caso de uso número 7: "Alta de congresos"

Resumen de la funcionalidad: el *Administrador*, podrá dar de alta los congresos del sistema.
Papel dentro del trabajo del usuario: es el caso de uso principal del actor *Administrador*.
Actores: Administrador.
Casos de uso relacionados: Login, Baja de congresos, Modificación de congresos y Consulta de congresos
Precondición: el congreso no tiene que existir y el usuario tiene que estar registrado e identificado.
Poscondición: el congreso queda dado de alta en el sistema.
El *Administrador,* entra en la opción mantenimiento de congresos y dentro de esta opción entra en la opción alta de congresos y rellena los datos de un congreso y lo da de alta en el sistema.

2.13.8.- Caso de uso número 8: "Baja de congresos"

Resumen de la funcionalidad: el *Administrador*, podrá dar de baja los congresos del sistema.
Papel dentro del trabajo del usuario: es un caso de uso ocasional del actor *Administrador*.
Actores: Administrador.
Casos de uso relacionados: Login y Alta de congresos.
Precondición: el congreso tiene que existir y el usuario tiene que estar registrado e identificado.
Poscondición: el congreso queda dado de baja en el sistema.
El *Administrador,* entra en la opción mantenimiento de congresos y dentro de esta opción entra en la opción baja de congresos e introduce el identificador del congreso a dar de baja y lo da de baja en el sistema.

2.13.9.- Caso de uso número 9: "Modificación de congresos"

Resumen de la funcionalidad: el *Administrador*, podrá modificar los congresos dados de alta en el sistema.
Papel dentro del trabajo del usuario: es un caso de uso principal del actor *Administrador*.
Actores: Administrador.
Casos de uso relacionados: Login y Alta de congresos
Precondición: el congreso tiene que existir y el usuario tiene que estar registrado e identificado.
Poscondición: el congreso queda modificado con los nuevos cambios en el sistema.
El *Administrador,* entra en la opción mantenimiento de congresos y dentro de esta opción entra en la opción modificación de congresos e introduce el identificador del congreso y a continuación modifica los datos que sean necesarios y los guarda quedando el congreso modificado en el sistema.

2.13.10.- Caso de uso número 10: "Alta de usuarios"

Resumen de la funcionalidad: el *Administrador*, podrá dar de alta los usuarios registrados del sistema para que puedan gestionar sus congresos y publicaciones.
Papel dentro del trabajo del usuario: es un caso de uso principal del actor *Administrador*.
Actores: Administrador.
Casos de uso relacionados: Login, Baja de usuarios, Modificación de usuarios y Consulta de usuarios.
Precondición: el usuario no tiene que existir y el *Administrador* tiene que estar registrado e identificado.
Poscondición: el usuario queda dado de alta en el sistema.
El *Administrador,* entra en la opción mantenimiento de usuario, y dentro de esta opción entra en la opción alta de usuarios y rellena los datos de un usuario y lo da de alta en el sistema.

2.13.11.- Caso de uso número 11: "Baja de usuarios"

Resumen de la funcionalidad: el *Administrador*, podrá dar de baja los usuarios registrados del sistema.
Papel dentro del trabajo del usuario: es un caso de uso ocasional del actor *Administrador*.
Actores: Administrador.
Casos de uso relacionados: Login y Alta de usuarios.
Precondición: el usuario tiene que existir y el *Administrador* tiene que estar registrado e identificado.
Poscondición: el usuario queda dado de baja en el sistema.
El *Administrador,* entra en la opción mantenimiento de usuarios y dentro de esta opción entra en la opción baja de usuarios e introduce el identificador del usuario a dar de baja y lo da de baja en el sistema.

2.13.12.- Caso de uso número 12: "Modificación de usuarios"

Resumen de la funcionalidad: el *Administrador*, podrá modificar los usuarios dados de alta en el sistema.
Papel dentro del trabajo del usuario: es un caso de uso esporádico del actor *Administrador*.
Actores: Administrador.
Casos de uso relacionados: Login y Alta de usuarios
Precondición: el usuario tiene que existir y el *Administrador* tiene que estar registrado e identificado.
Poscondición: el usuario queda modificado con los nuevos cambios en el sistema.
El *Administrador,* entra en la opción mantenimiento de usuarios y dentro de esta opción entra en la opción modificación de usuarios e introduce el identificador del usuario y a continuación modifica los datos que sean necesarios y los guarda quedando el usuario modificado en el sistema.

2.13.13.- Caso de uso número 13: "Consulta de usuarios"

Resumen de la funcionalidad: Busca el usuario que el *Administrador* ha seleccionado mediante una función de búsqueda.
Papel dentro del trabajo del usuario: es un caso de uso ocasional en el *Administrador*.
Actores: Administrador.
Casos de uso relacionados: Alta de usuarios y Login
Precondición: El usuario tiene que estar dado de alta en el sistema
Poscondición: ninguna, es una consulta
Busca el *usuario* que cumple con la condición especificada en el criterio de búsqueda, y se obtiene una ficha del usuario en cuestión.

2.13.14.- Caso de uso número 14: "Login"

Resumen de la funcionalidad: Identificación de los usuarios registrados para acceder a funcionalidades reservadas a los mismos.
Papel dentro del trabajo del usuario: es un caso de uso principal en el *Profesor gestor y Administrador.*
Actores: Administrador, Profesor gestor.
Casos de uso relacionados: con todos los casos de uso relacionados con los actores *Administrador* y *Profesor gestor.*
Precondición: Tienen que estar dados de alta en el sistema, registrados.
Poscondición: Los usuarios quedan identificados en el sistema, pudiendo acceder a las funcionalidades específicas para los usuarios registrados.
Proceso por el cual un usuario introduce el login y password y a continuación acepta los datos, el proceso identifica o no al usuario y le da privilegios de usuario registrado, si la identificación es correcta.

2.14.- La interfaz de usuario

Recoger información y requisitos y documentarlos de la interfaz de usuario consiste en documentar y verificar información sobre los usuarios, su trabajo actual y su visión de su trabajo con la aplicación futura.

La interfaz de usuario es aquello que los usuarios ven del funcionamiento del programa. Dependen los siguientes factores:

- Comodidad del usuario
- Productividad del usuario
- Imagen del programa

2.15.- Identificación de las restricciones técnicas

Es necesario determinar si se ha establecido el uso de alguna tecnología, herramienta, plataforma o normativa, para el desarrollo del programa.

Separación por capas J2EE, utilizando bases de datos relacionales, web accesibles y utilización de HTML.

Elaboración de los perfiles de los usuarios

No se puede diseñar bien una interfaz de usuario sin saber para quien se diseña. Los perfiles de usuarios tienen estos aspectos:

- Experiencia en el entorno de hardware y software.
- Experiencia en aplicaciones del mismo dominio.
- Experiencia en el trabajo.
- Frecuencia de uso del programa.

Los usuarios *Profesor gestor* y *Administrador*, conocen a fondo su trabajo. Y están acostumbrados a utilizar ordenadores, pero tengo que tener cuidado a la hora de diseñar la interfaz de usuario para los *Invitados*, ya que éstos pueden que no tengan mucha experiencia a la hora de navegar por Internet o en el manejo de los ordenadores. Todas las funciones de la aplicación se utilizarán muy a menudo.

3.- Análisis

3.1.- Introducción

El análisis de la aplicación seguirá las técnicas orientadas a objetos, aplicando las notaciones y conceptos de UML y siguiendo el ciclo de vida del Racional Unified Process. El análisis será la primera etapa del desarrollo de la aplicación propiamente dicha. Esta fase consistirá en traducir los requisitos a una forma más adecuada para ser la base de partida de este desarrollo, para obtener el modelo del análisis, que se irá actualizando y se utilizará en todo el ciclo de vida.

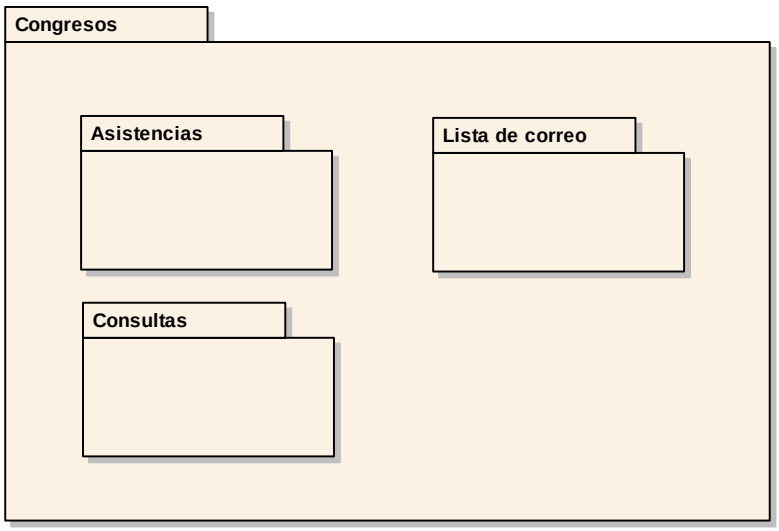
Después de la etapa de recogida y documentación de requisitos, tenemos que identificar las clases fundamentales y expresar los casos de uso en términos de estas clases. En esta etapa se describirá el funcionamiento de la futura aplicación.

3.2.- Paquetes de análisis y paquetes de servicios

Dividiremos la documentación en paquetes de UML, cada uno de los cuales contendrá clases, casos de uso u otros elementos de UML. Los paquetes siempre cumplirán las condiciones de ser coherentes y pocos dependientes.

Los paquetes de análisis corresponden a uno o diversos subsistemas enteros, mientras que los paquetes de servicios son subdivisiones de los paquetes de análisis desde un punto de vista comercial, desde el punto de vista cliente.

En nuestro caso solamente se ve un único paquete de análisis, *Congresos* que comprenderá todos los casos de uso analizados con anterioridad. El esquema sería:



3.3.- Revisión de los casos de uso

La base de partida para el análisis es la documentación sobre los casos de uso elaborada en la etapa anterior. En esta fase repaso los casos de uso viendo que de momento el diagrama es correcto.

Pero conviene tener presente que en la aplicación de Congresos la información se tiene que poder modificar, que toda la información se tiene que poder consultar y que raramente hay información eterna, se tiene que ir borrando aquella que ya no haga falta, a un fichero de histórico por ejemplo.

3.4.- Especificación

En este apartado se llevará a cabo la especificación de las clases del análisis.

Tenemos que distinguir tres tipos de clases de análisis:

1. Clases de frontera: representan la interfaz de usuario por pantalla. Tiene que haber al menos una para cada papel de cada actor. Cada una representa la interfaz de usuario entre un caso de uso y un actor.

2. Clases de entidad: corresponden a los objetos del dominio, modelan entidades o acontecimientos del mundo real, muchos tendrán que ser persistentes lo que obligará a almacenarlos en nuestra base de datos MySQL.
3. Clases de control: corresponden a objetos internos de la aplicación y no persistentes. Contienen la parte principal de los algoritmos de la aplicación.

3.5.- Identificación de las clases de entidad

Consiste en identificar unas primeras clases mediante las cuales se puedan especificar los casos de uso en forma de interacciones.

Muchas de estas clases pasarán al diseño y a la implementación. Buscamos clases de conjuntos de objetos homogéneos, que tienen unos mismos atributos y operaciones.

Identificaré las clases de entidad a partir de los casos de uso. Para cada caso de uso se indican las clases que se encuentran, cuando una clase ya se había identificado en un caso de uso anterior, su nombre irá seguido de un asterisco, mientras que las clases que son dudosas irán seguidas de un interrogante.

Caso de uso número 1: "Consultar congreso". Clases: *Usuario, Invitado?, Profesor?, Administrador?, Congreso, Calendario?, Ficha?*
Caso de uso número 2: "Apuntarse a lista de correo". Clases: *Lista_correo, Congreso*,Usuario*.*
Caso de uso número 3: "Darse de baja de la lista de correo". Clases: *Usuario*, Lista_correo*, Congreso*.*
Caso de uso número 4: "Apuntarse a un congreso". Clases: *Usuario*, Congreso*, Viaje?, Publicacion?, Asistencia.*
Caso de uso número 5: "Darse de baja de un congreso". Clases: *Usuario*, Congreso*, Viaje?*, Publicacion?*, , Asistencia*.*
Caso de uso número 6: "Enviar documentación". Clases: *Publicacion?*, Congreso*, Usuario*.*
Caso de uso número 7: "Alta congresos". Clases: *Congreso*, Usuario*.*
Caso de uso número 8: "Baja de congresos". Clases: *Congreso*, Usuario*.*
Caso de uso número 9: "Modificación de congresos". Clases: *Congreso*, Usuario*.*
Caso de uso número 10: "Alta de usuarios". Clases: *Usuario*, Congreso*, Publicacion*.*
Caso de uso número 11: "Baja de usuarios". Clases: *Usuario*.*
Caso de uso número 12: "Modificación de usuarios". Clases: *Usuario*.*
Caso de uso número 13: "Consulta de usuarios". Clases: *Usuario*.*
Caso de uso número 14: "Login". Clases: *Usuario*, Congreso*, Publicacion?*, Viaje?*.*

Por tanto la primera lista de clases de entidad es: *Usuario, Congreso, Lista_correo, Publicacion, Asistencia, Role* y *Tema.*

He descartado *Invitado, Profesor* y *Administrador*, ya que son actores y para la aplicación en sí creo que no son relevante, tan solo la identificación contra la

aplicación será significativa por eso me basta con una clase *Usuario,* además ya que los usuarios deben tener una temática, creo una clase tipificada que me evitará errores futuros a la hora de escoger la categoría de los usuarios.

También he descartado las clases *Calendario* y *Ficha*, ya que no tienen que crearse ningún objeto de ellas y por consiguiente no deberé tener que guardar ninguno de ellos, ya que no existirán, por eso no tienen persistencia.

He descartado *Viaje*, ya que creo que no tenemos que guardar ninguna información relevante de ellos, sino que la misma web del Congreso contendrá la información del viaje en sí.

Por el mismo motivo que la clase *Role*, he creado la clase *Tema*, ya que los congresos son de una temática determinada y para no cometer errores en futuras consultas he tipificado los temas.

3.6.- Especificación de los atributos de las clases de entidad

Los atributos de las clases de entidad son generalmente datos nombrados explícitamente en los casos de uso. Los usuarios los usan directamente.

Clase *Usuario: dni*(string), *nombre*(string), *ape1*(string), *ape2*(string), *role*(integer), *login*(string), *password*(string).
Clase *Congreso: id*(string), *acronimo*(string), *edicion*(integer), *tematica*(string), *fechainí*(date), *fechafin*(date), *denominacion*(string), *direccion*(string), *ciudad*(string), *pais*(string), *web*(string), *organizador*(string), *id_lista_correo_fk*(integer), *viaje*(string), *descripcion*(string), *lengua_vehicular*(string), *precio*(float), *feciniins*(date), *fecfinins*(date), *feciniabstract*(date), *fecfinabstract*(date), *fecaceptacion*(date), *fax*(string), *email*(string), *telefono*(string), *cp*(string).
Clase *Asistencia: id*(integer), *dni_usuario_fk*(string), *id_congreso_fk*(string), *ponente*(boolean), *npublicaciones*(integer).
Clase Publicacion: *id*(integer), *titulo*(string), *autores*(string), *aceptado*(boolean) *id_asistencia_fk*(integer).
Clase Lista_correo: *id*(integer), *dni_usuario*(string), *email*(string), *congresos[ ]*(integer).
Clase Tema: *id*(integer), *descripcion*(string).
Clase Role: *id*(integer), *descripcion*(string).

3.7.- Identificación de las relaciones entre clases

Ahora ya tenemos la lista completa de las clases de la aplicación. Ahora es el momento de examinar las relaciones de herencia, asociaciones y las relaciones de agregación.

Jerarquía de herencia

Establecemos las jerarquías de herencia por dos vías, mediante la especialización y la generalización

Herencia múltiple

Se da cuando una clase tiene diversas superclases y hereda atributos, operaciones, asociaciones y agregaciones de todas ellas. Esto puede dar problemas al darse el caso de herencia repetidas, en nuestro caso al utilizar Java no se puede utilizar la herencia múltiple aunque sí simular.

Interficies

Equivalen a clases abstractas sin atributos y con todas las operaciones abstractas.

Asociaciones

Existirá una asociación entre clases cuando los objetos de una necesitan la colaboración de objetos de la otra para llevar a cabo sus operaciones. Las asociaciones pueden tener nombre y entre dos clases o más, puede haber más de una asociación con significado diferente, se puede dar el caso de existir una asociación entre una clase y ella misma. Si la asociación tiene atributos derivará en una clase asociativa. En la aplicación de Congresos, nos encontramos con una relación de asociación entre las clases *Usuario* y *Congreso*, ya los objetos de *Congreso*, necesitan la colaboración de los objetos de *Usuario* para llevar a cabo sus operaciones, decimos que el *Usuario <<asiste>>* a *Congresos.* De esta *asistencia,* debemos guardar muchos datos, ya que son importantes para la aplicación, por ello se crea la clase asociativa *Asistencia*. Existe otra asociación entre las clases *Asistencia* y *Publicacion,* decimos que en una *Asistencia <<se puede entregar>> Publicaciones.* Por último observo otra asociación entre las clases *Lista_correo* y *Congreso,* ya que el *Congreso <<tiene>>* una *Lista_correo* para que los usuarios se puedan apuntar.

Agregaciones

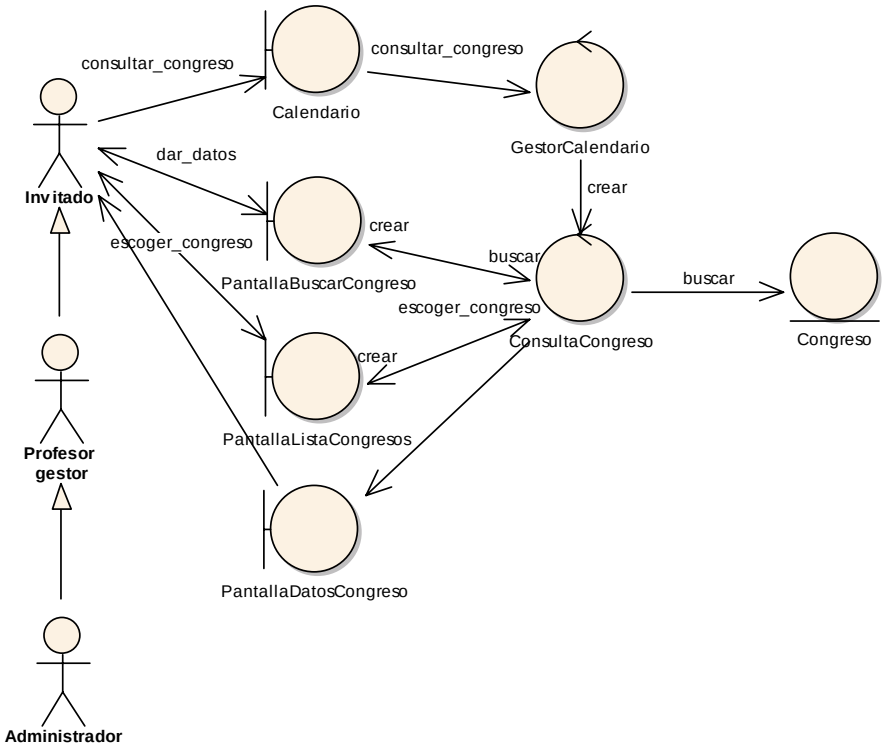
Caso particular de la asociación, en el que el significado de la asociación es que un papel es parte del otro en algún sentido y conviene aplicarlo siempre que nos encontremos en casos de este tipo.

3.8.- Identificación de las clases de frontera, las clases de control y de las operaciones.

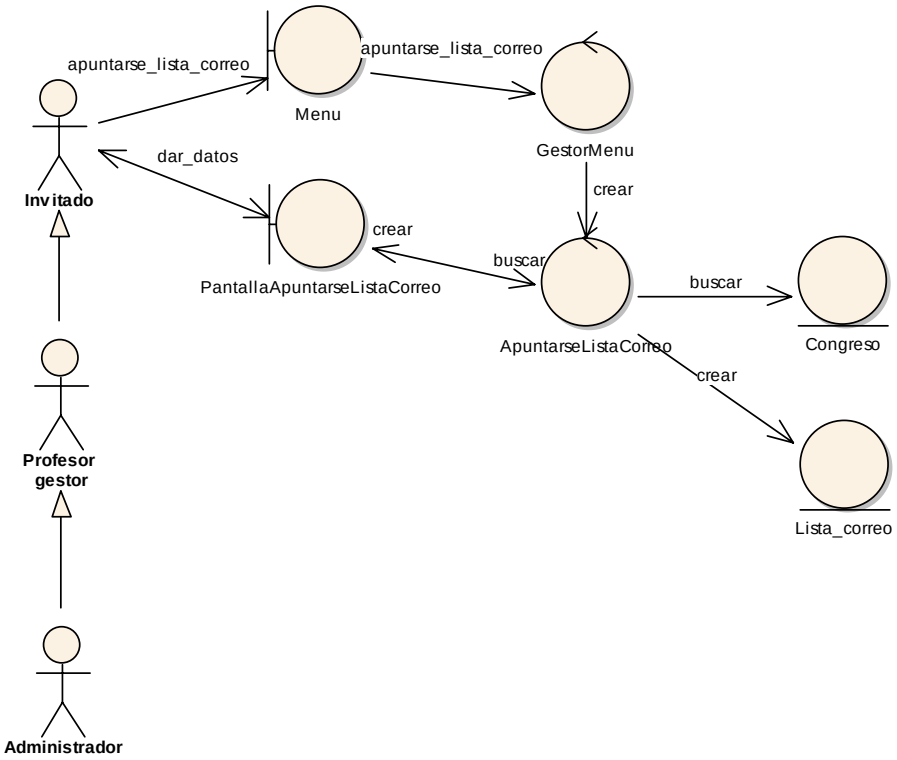
De la descripción textual de los casos de uso salen las operaciones de las clases de frontera y las de las clases de control.

Una manera de buscar y documentar las clases de frontera, control y las operaciones de los tres tipos de clase es elaborar diagramas de colaboración simplificados de los casos de uso representando las clases de los tres tipos. Después conviene hacer un diagrama estático en el cual figuren todas las clases de los tres tipos, con los atributos y operaciones identificadas y con todas las relaciones entre ellas, es decir el diagrama estático del análisis.

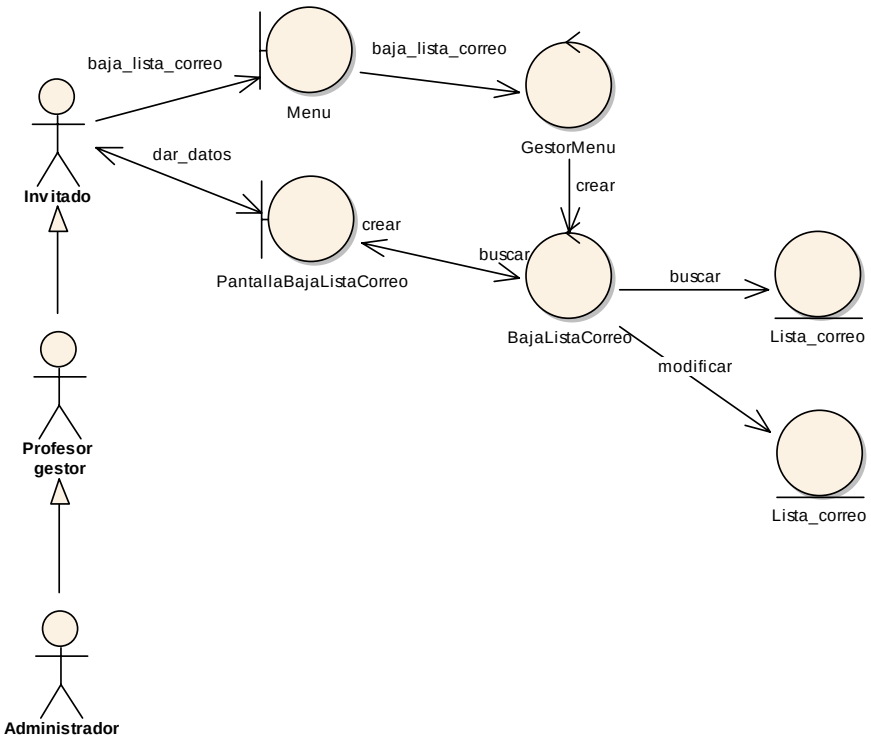
3.8.1.- Caso de uso número 1: "Consultar congreso"



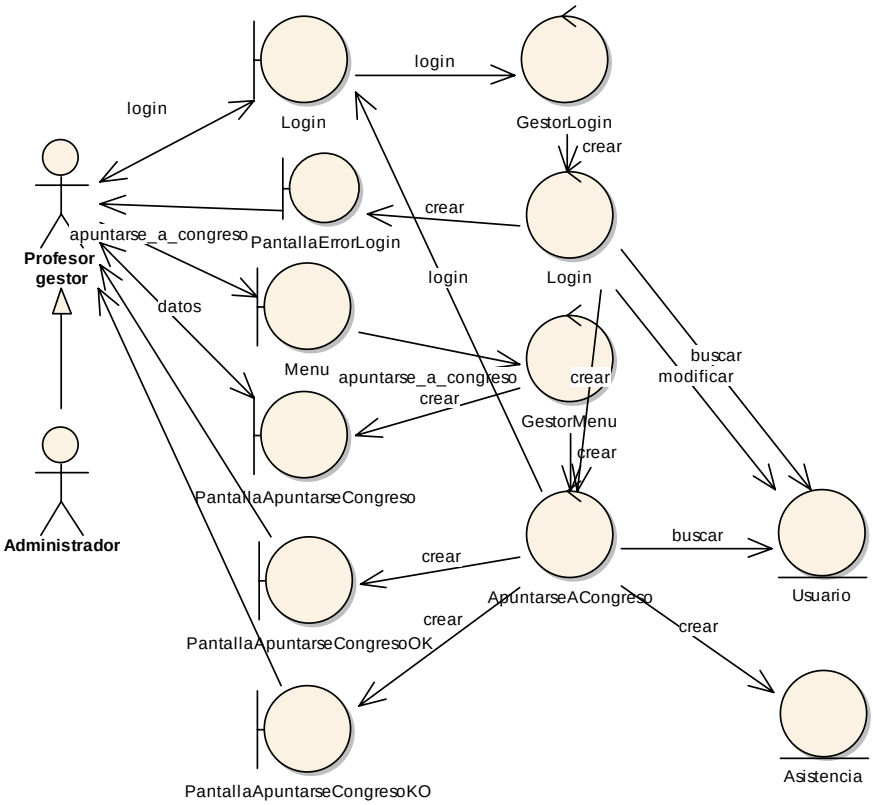
3.8.2.- Caso de uso número 2: "Apuntarse a lista de correo"



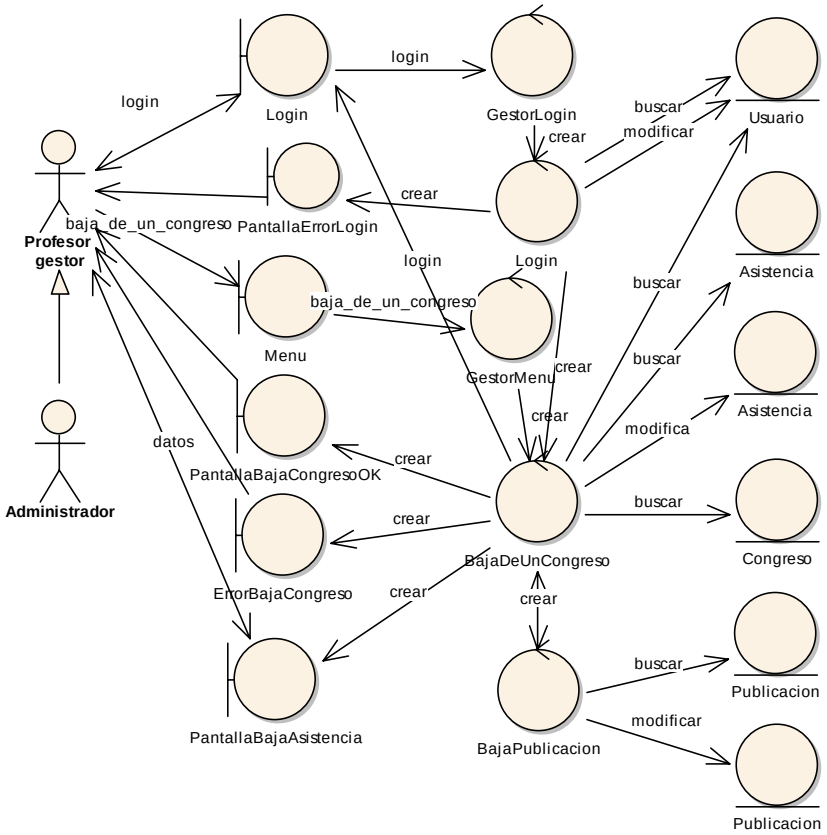
3.8.3.- Caso de uso número 3: "Darse de baja de la lista de correo"



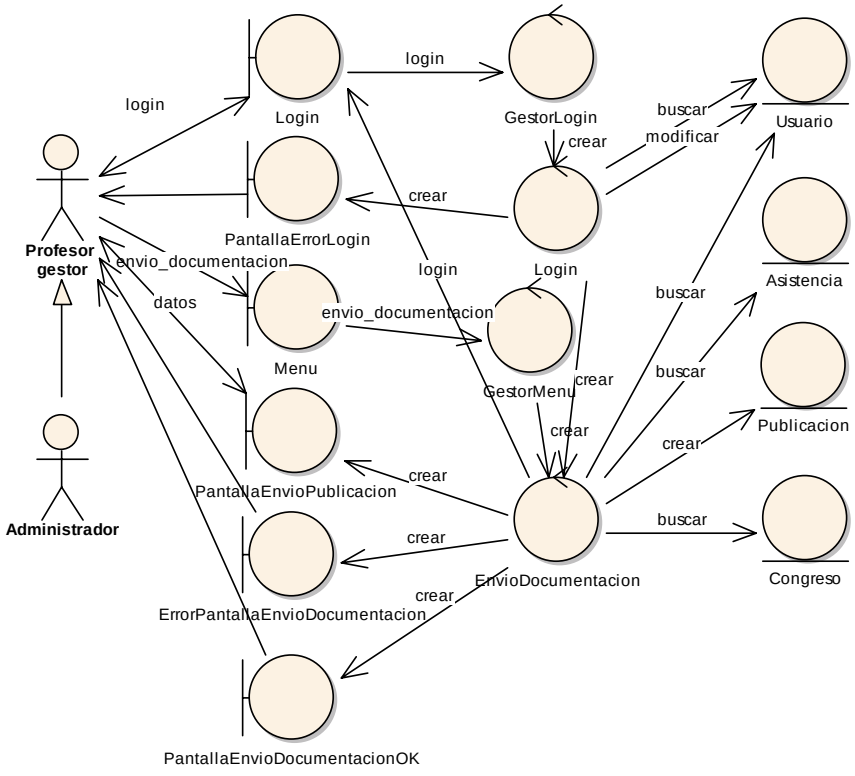
3.8.4.- Caso de uso número 4: "Apuntarse a un congreso"



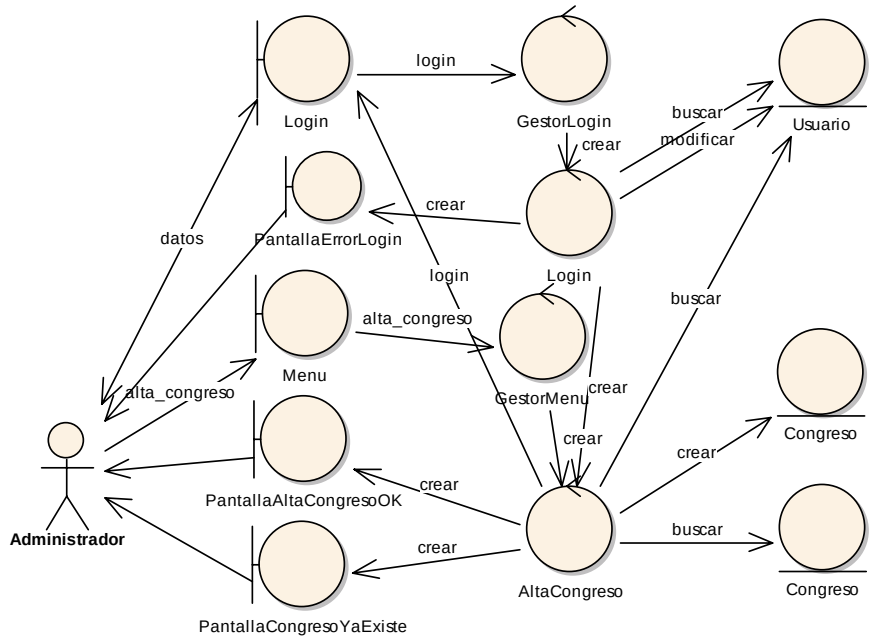
3.8.5.- Caso de uso número 5: "Darse de baja de un congreso"



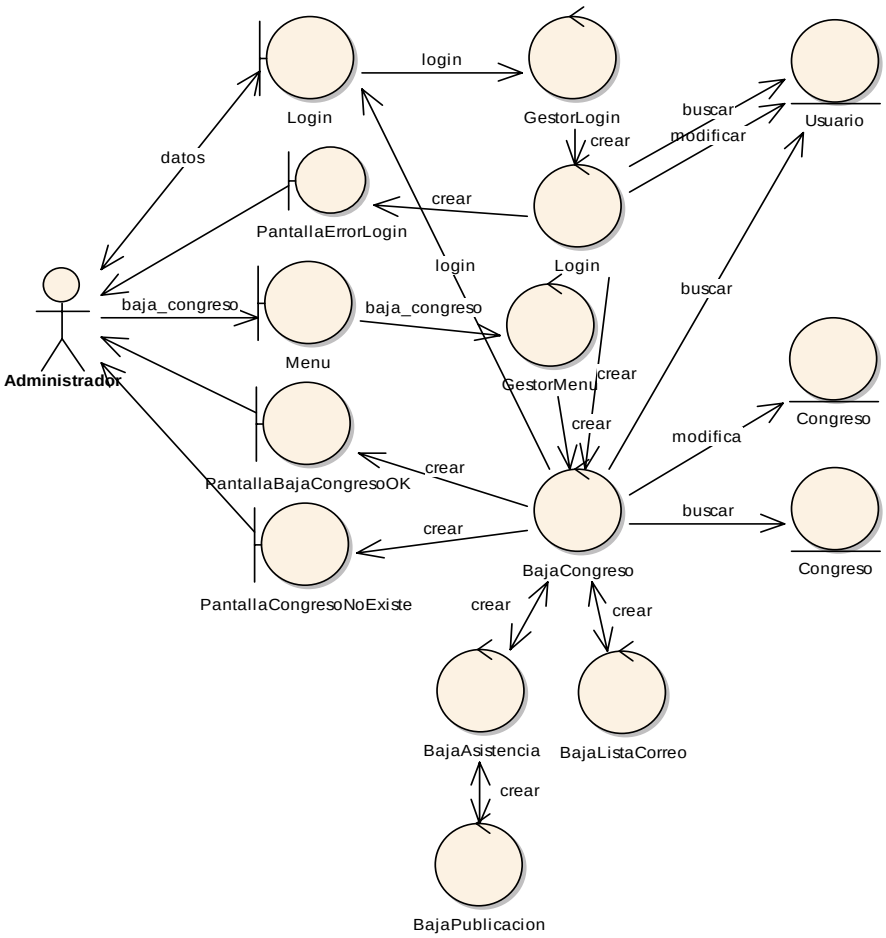
3.8.6.- Caso de uso número 6: "Enviar documentación"



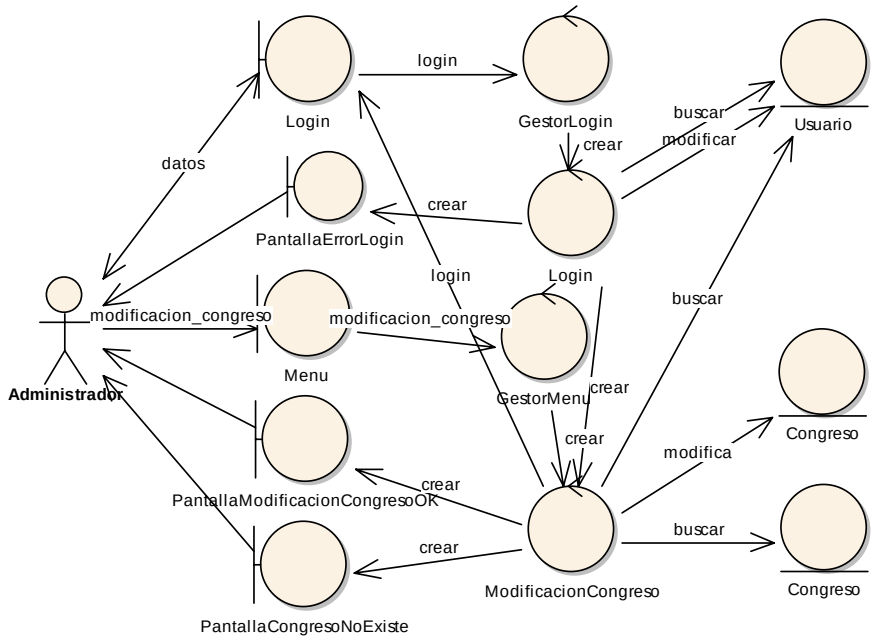
3.8.7.- Caso de uso número 7: "Alta de congresos"



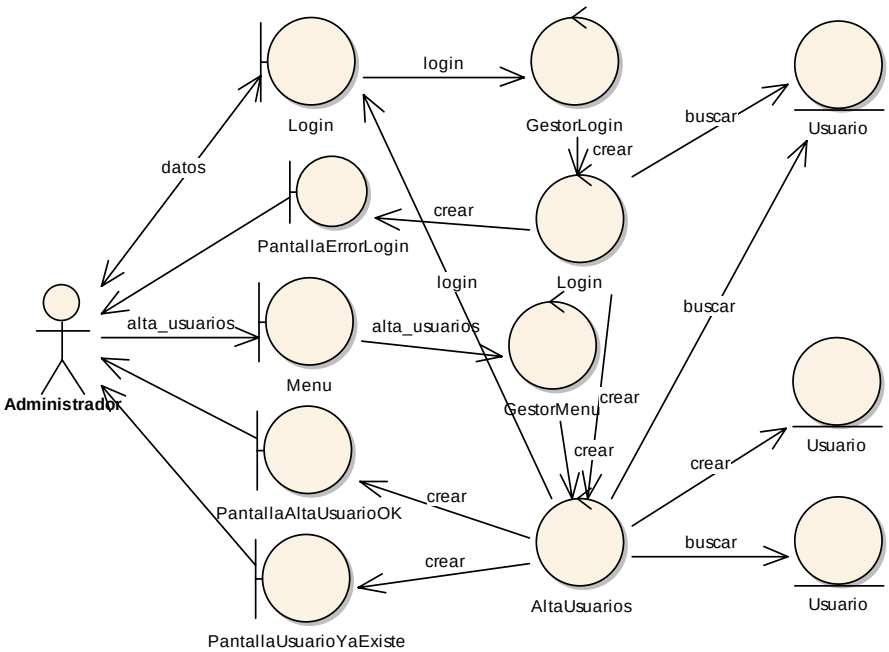
3.8.8.- Caso de uso número 8: "Baja de congresos"



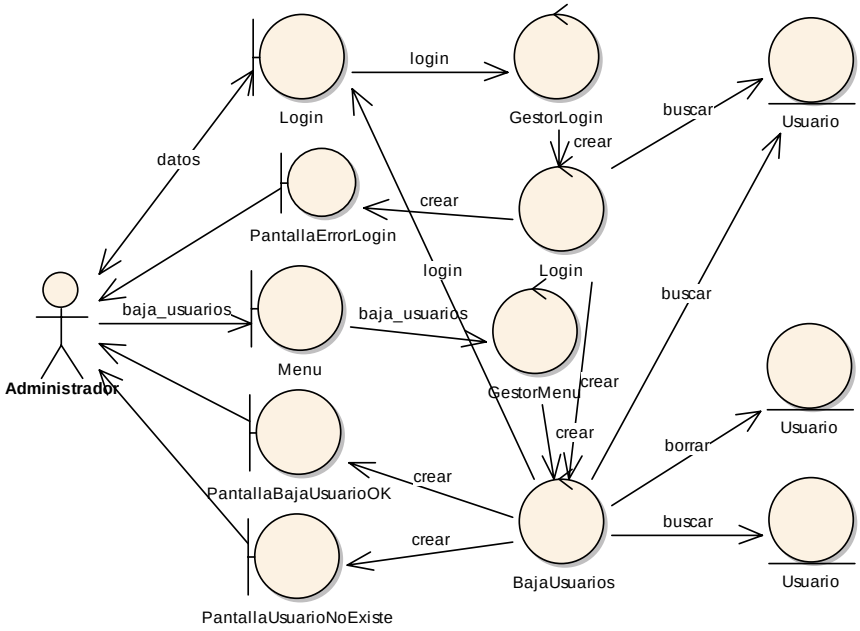
3.8.9.- Caso de uso número 9: "Modificación de congresos"



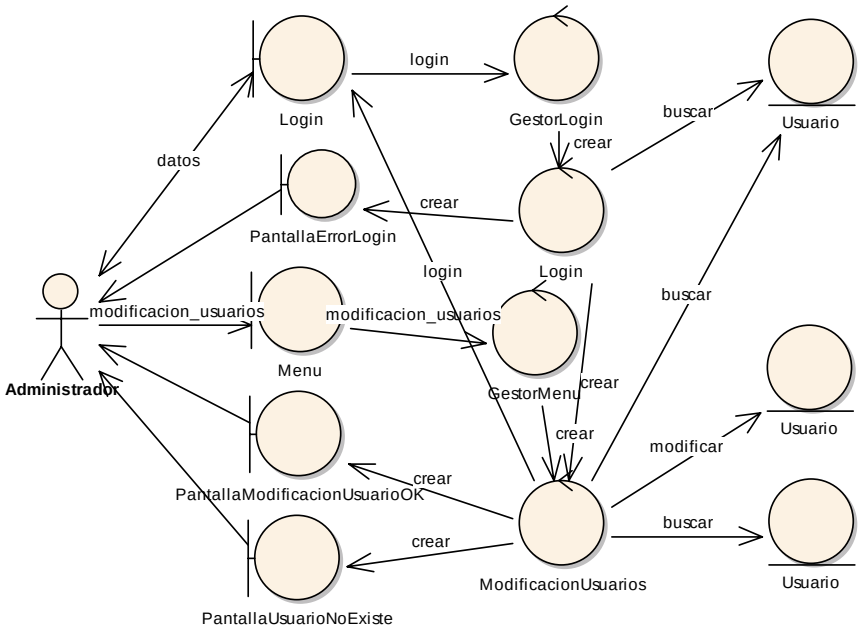
3.8.10.- Caso de uso número 10: "Alta de usuarios"



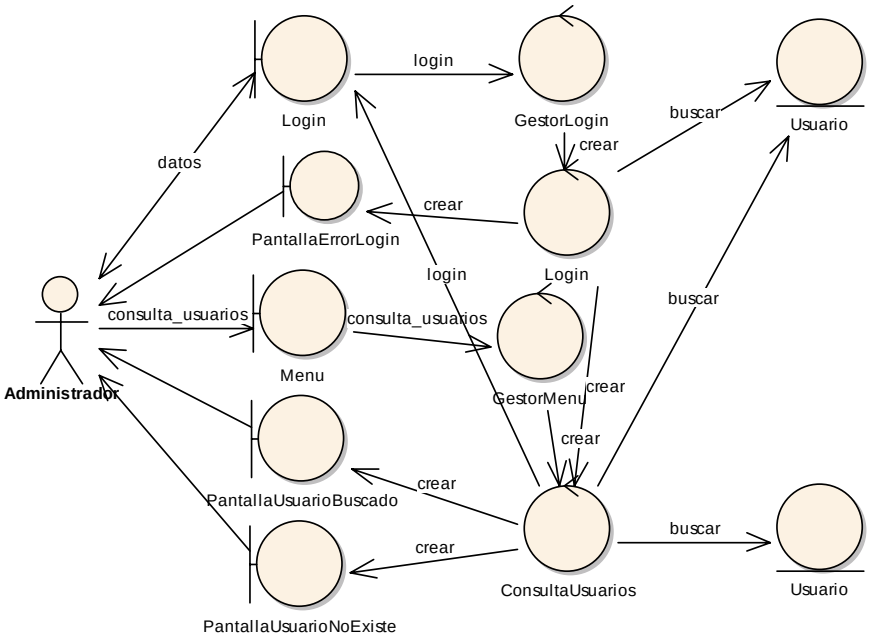
3.8.11.- Caso de uso número 11: "Baja de usuarios"



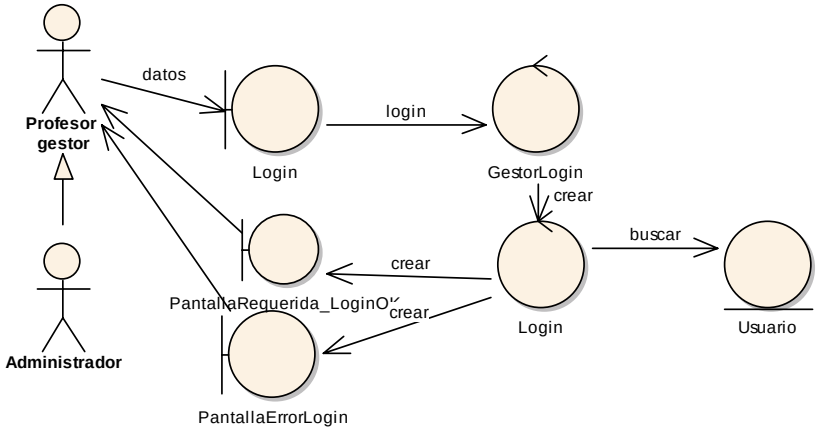
3.8.12.- Caso de uso número 12: "Modificación de usuarios"



3.8.13.- Caso de uso número 13: "Consulta de usuarios"

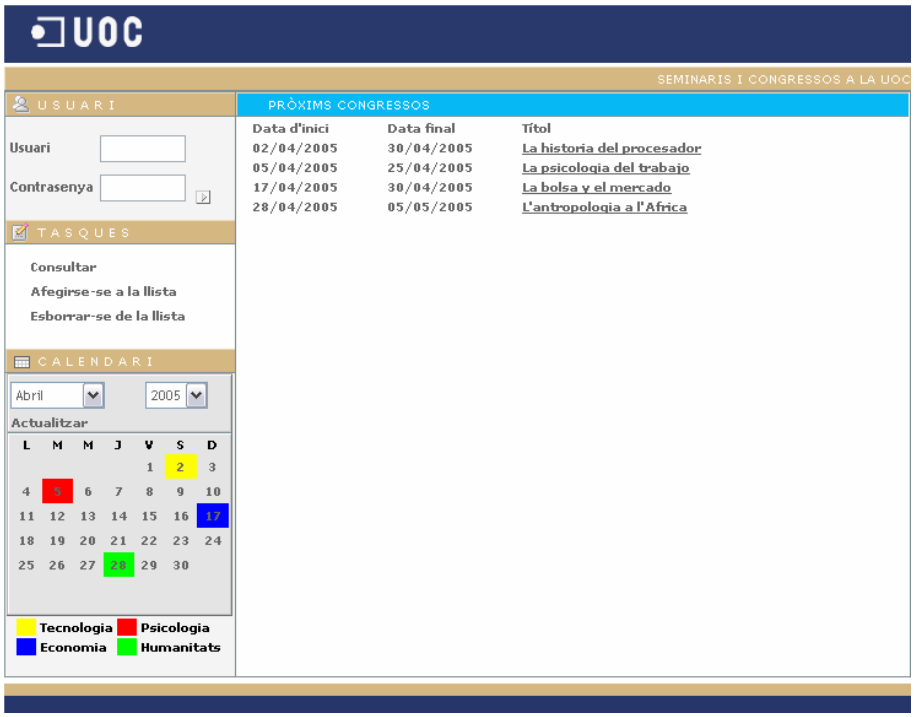


3.8.14.- Caso de uso número 14: "Login"

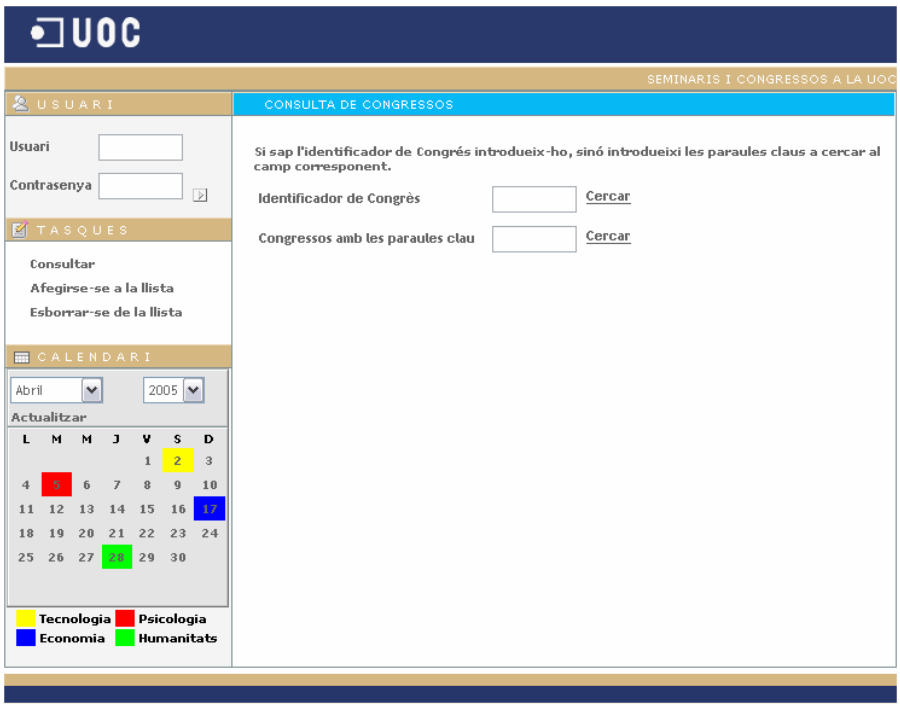


3.9.- Análisis de la interfaz de usuario

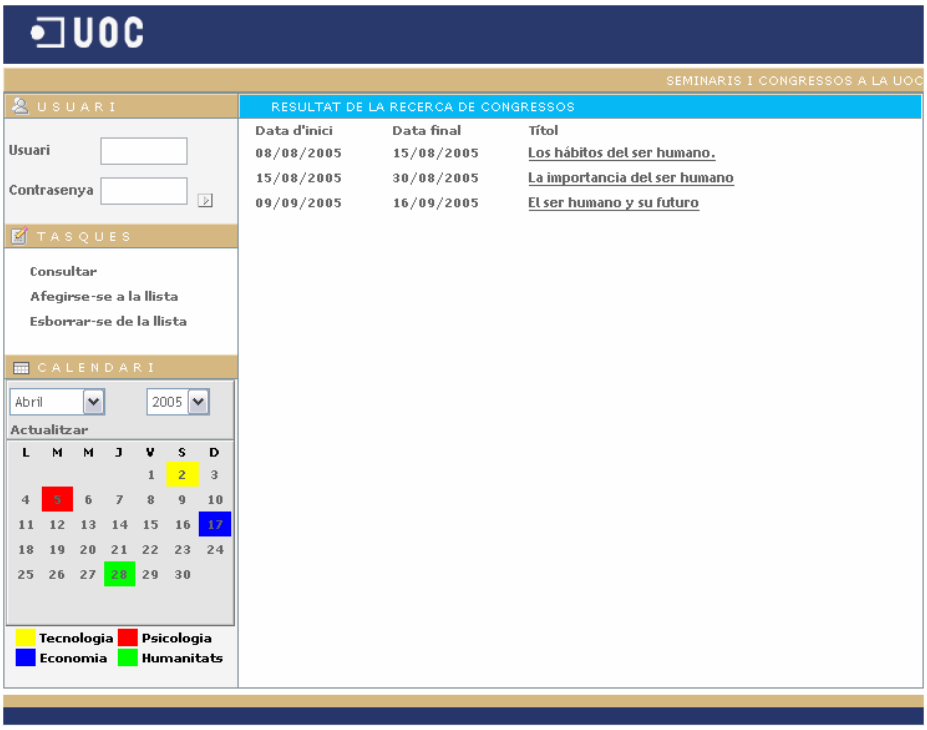
1) Clase Login / Menú / Calendario



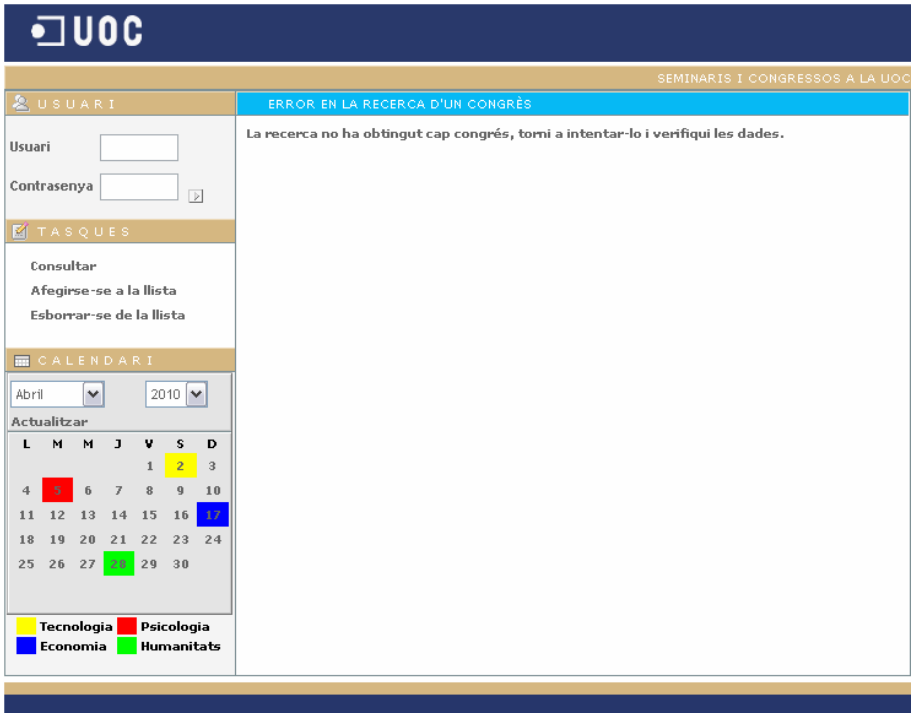
2) Clase PantallaBuscarCongreso



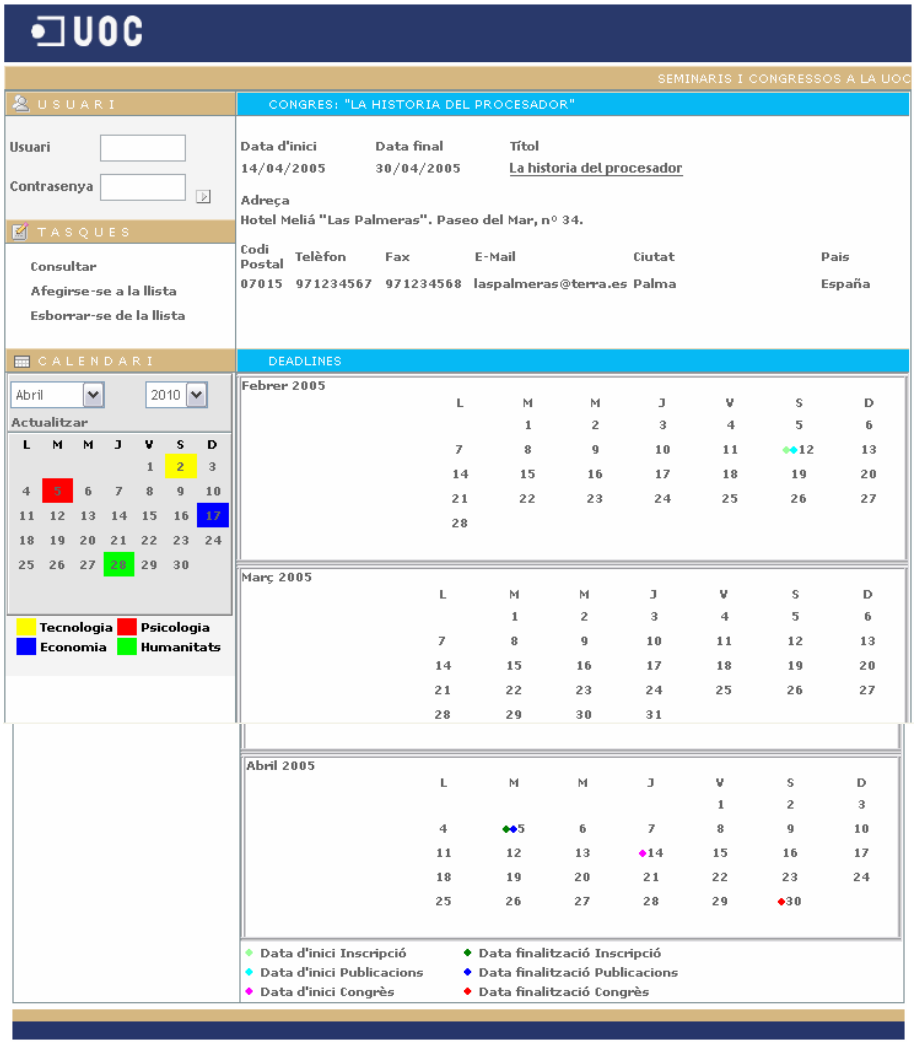
3) Clase PantallaListaCongresos



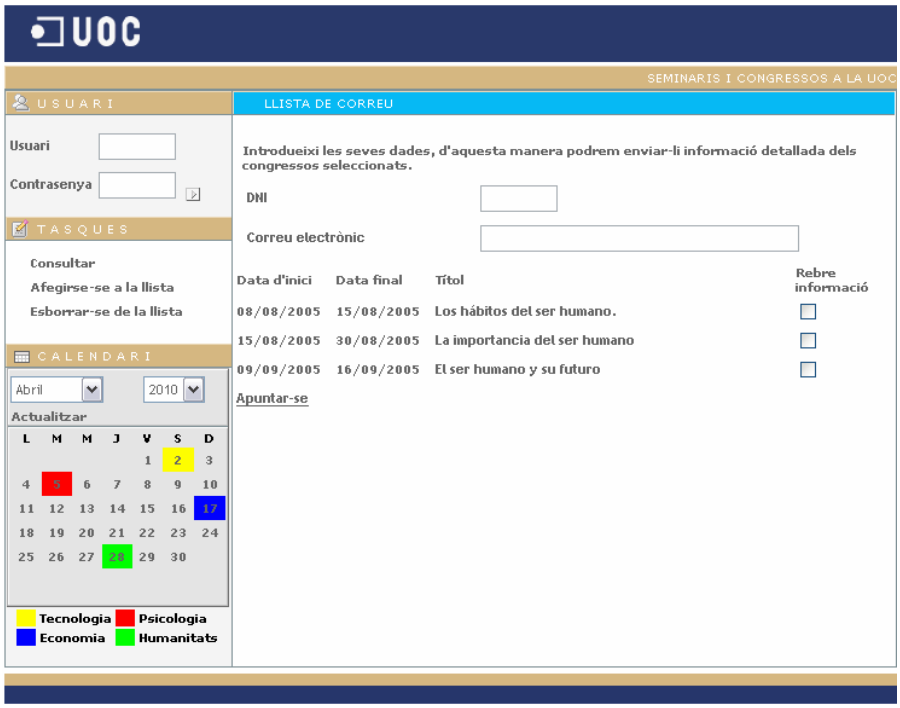
4) Clase ErrorBusquedaCongreso



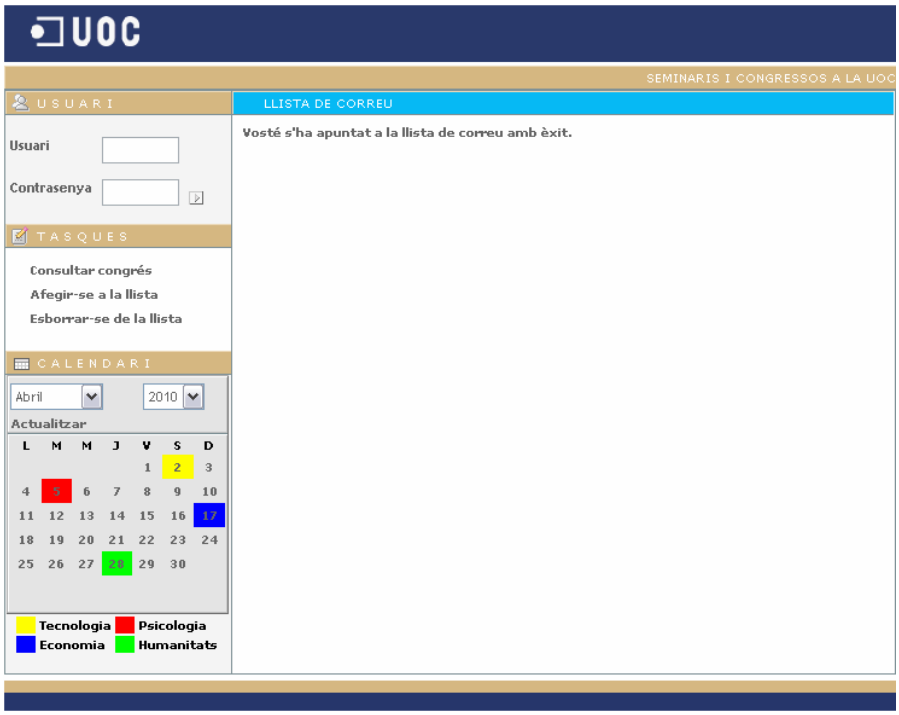
5) Clase PantallaDatosCongreso



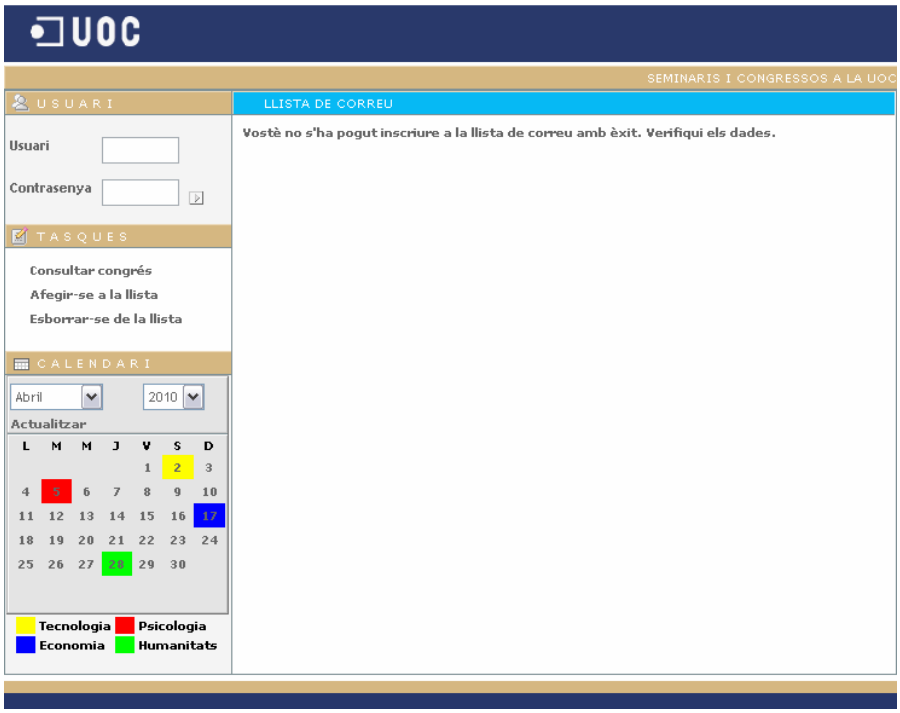
6) Clase PantallaApuntarseListaCorreo



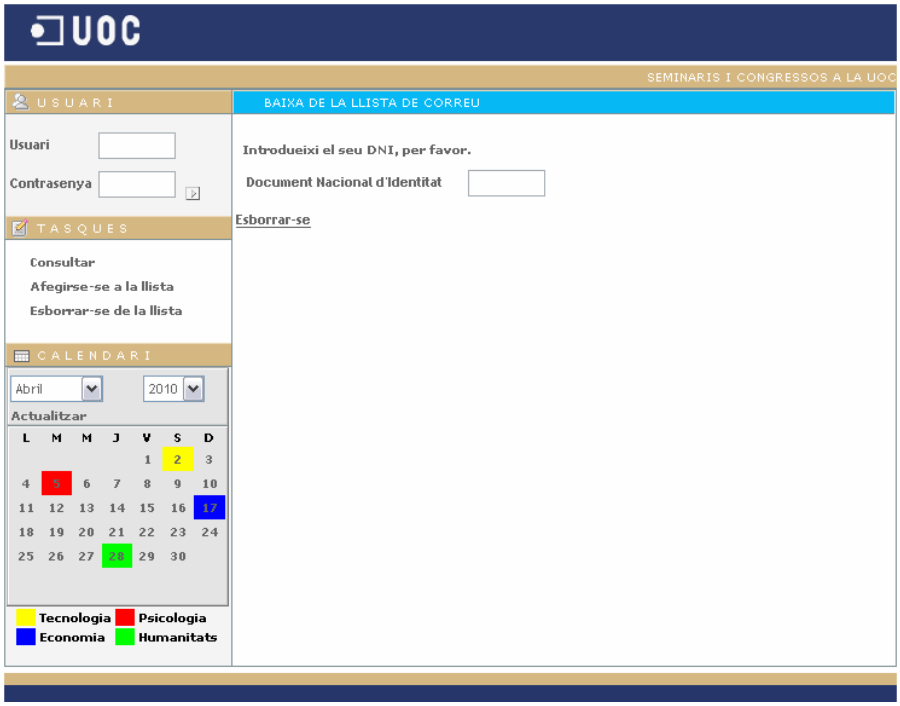
7) Clase PantallaApuntarseListaCorreoOk



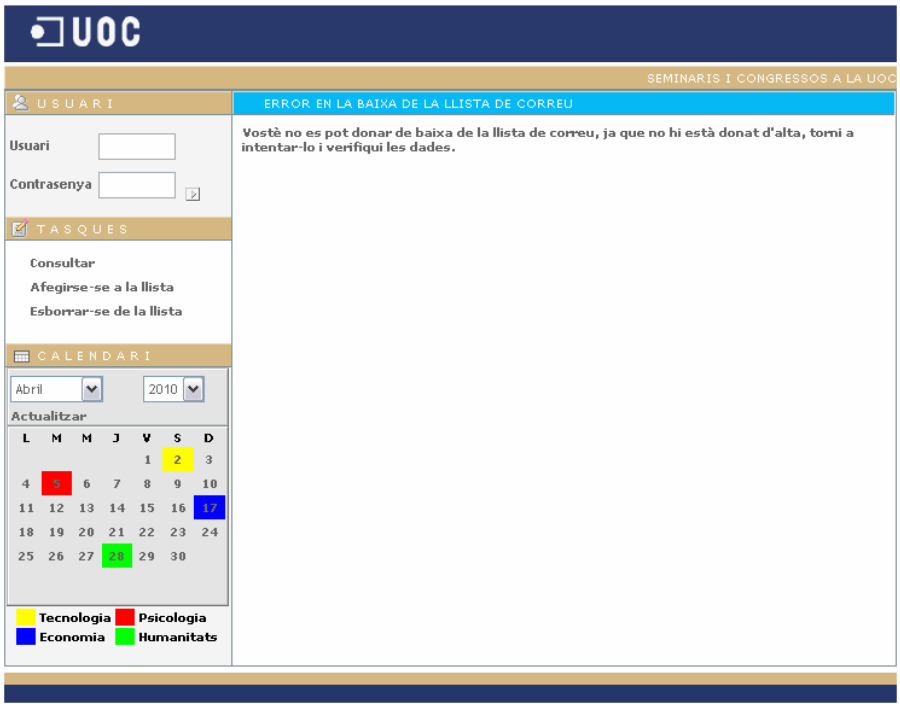
8) Clase ErrorApuntarseListaCorreo



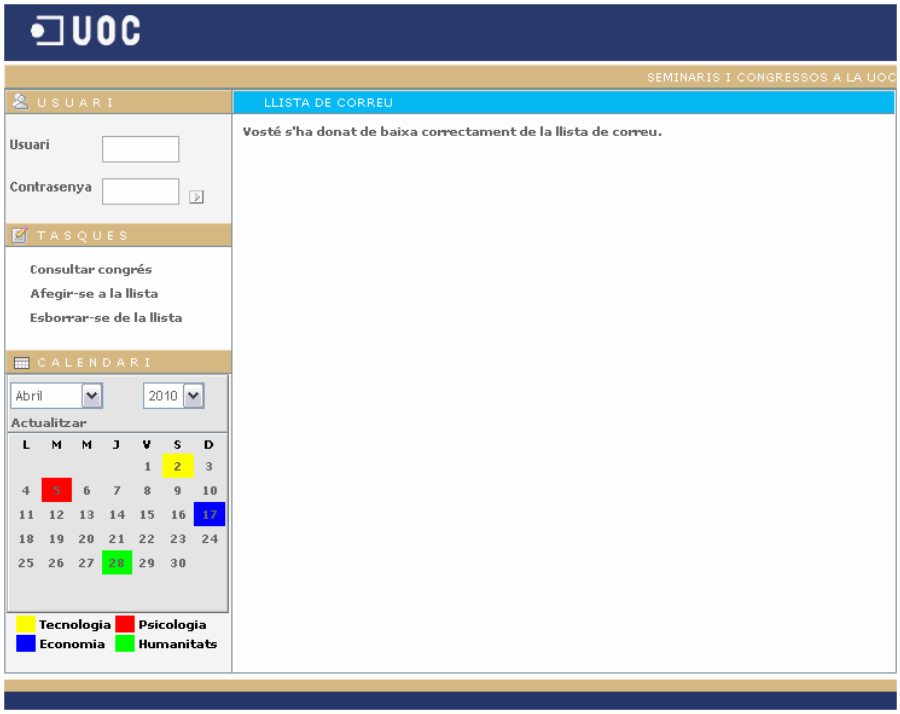
9) Clase PantallaBajaListaCorreo



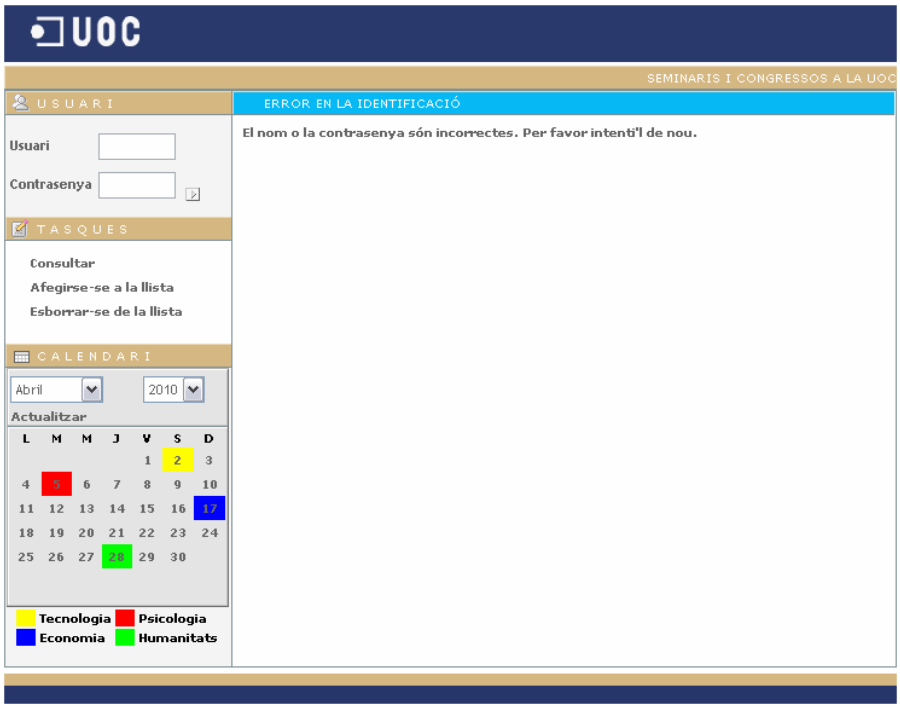
10) Clase ErrorBajaListaCorreo



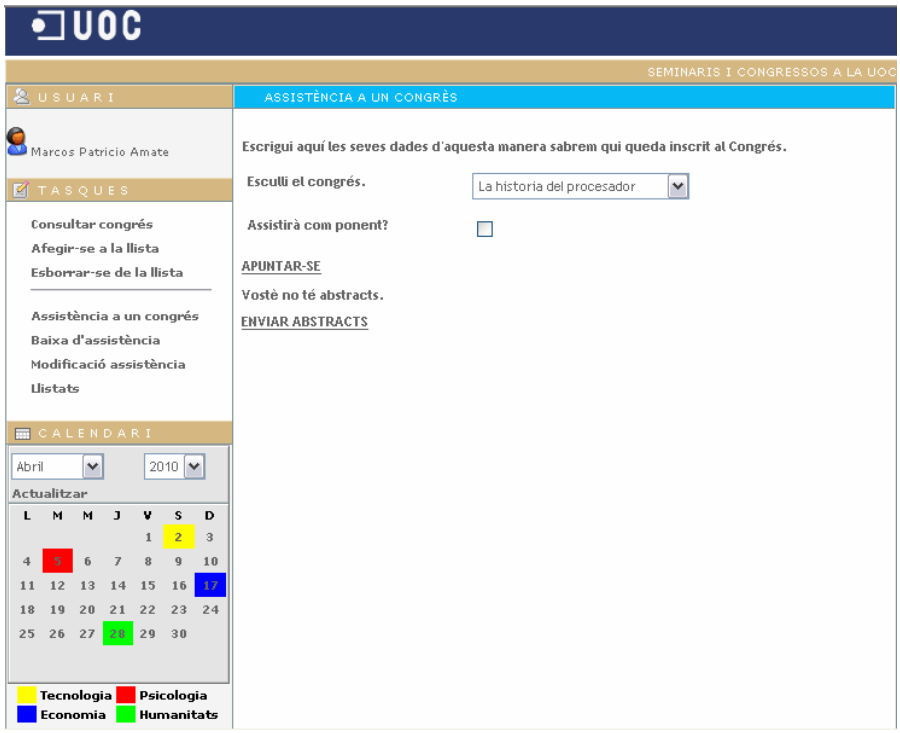
11) Clase PantallaBajaListaCorreoOk



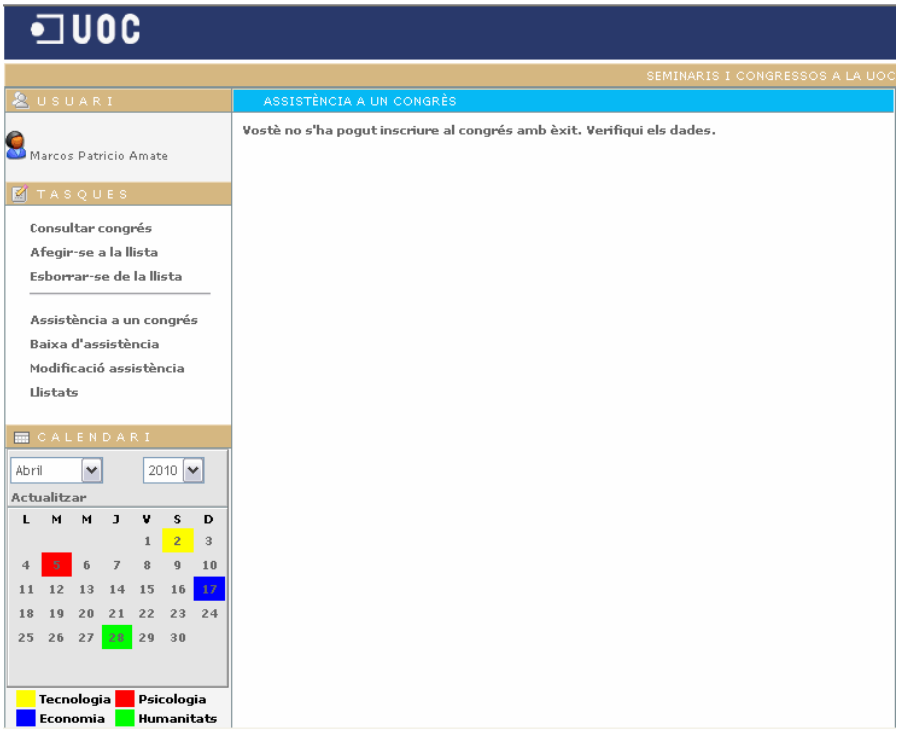
12) Clase ErrorLogin



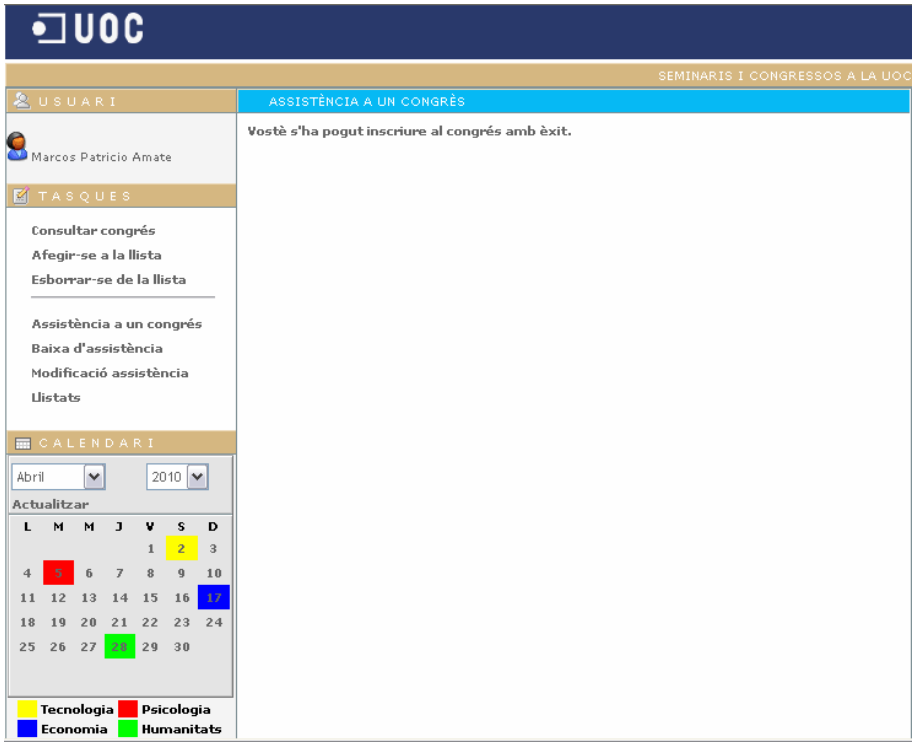
13) Clase ApuntarseCongreso



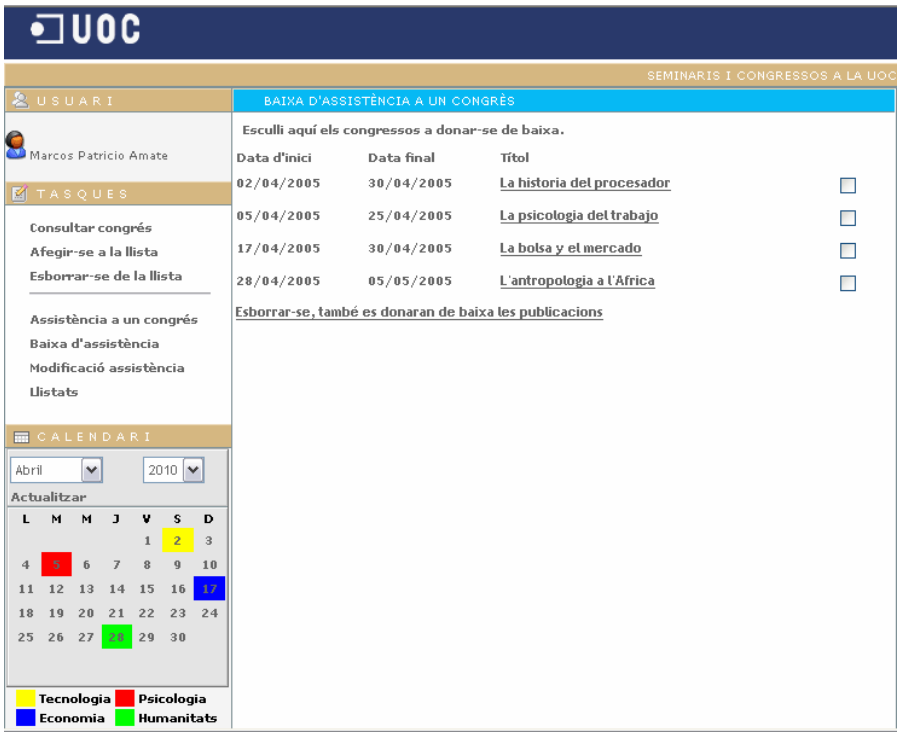
14) Clase ErrorApuntarseCongreso



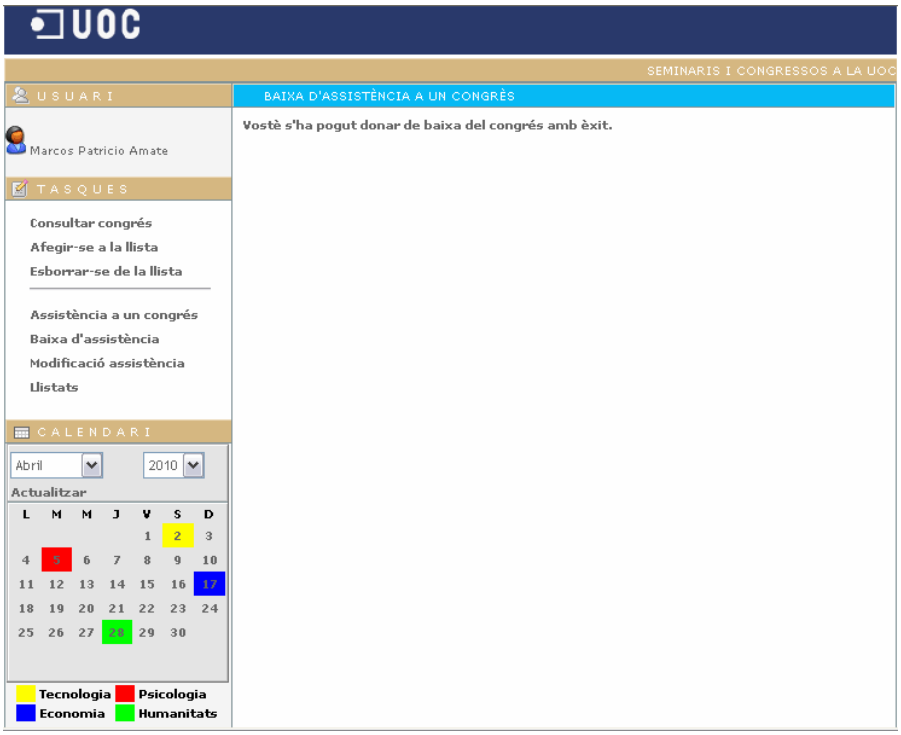
15) Clase ApuntarseCongresoOk



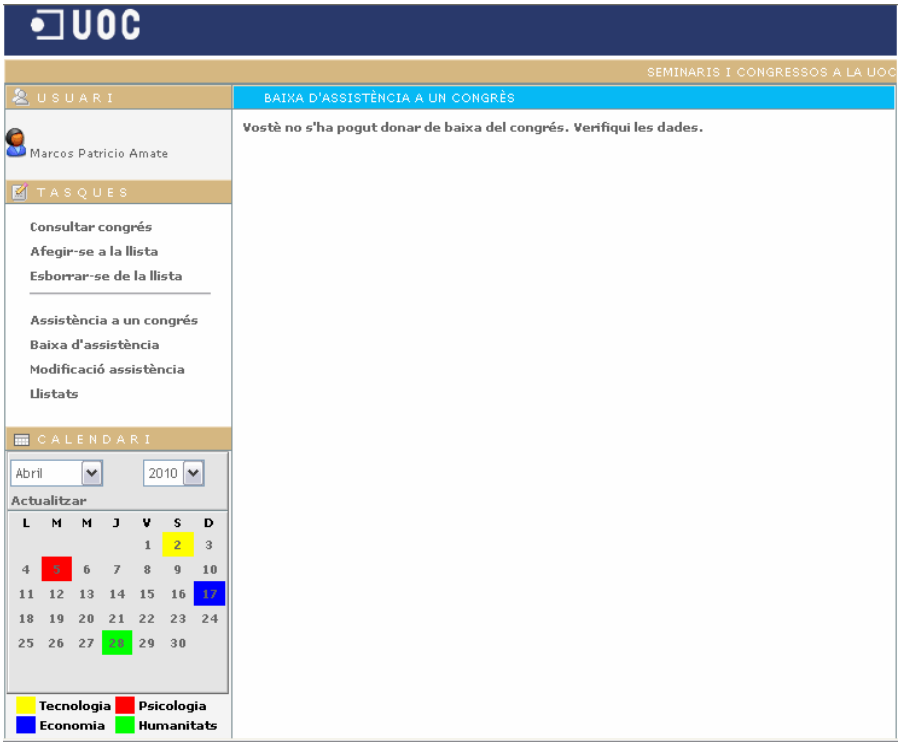
16) Clase PantallaBajaAsistencia



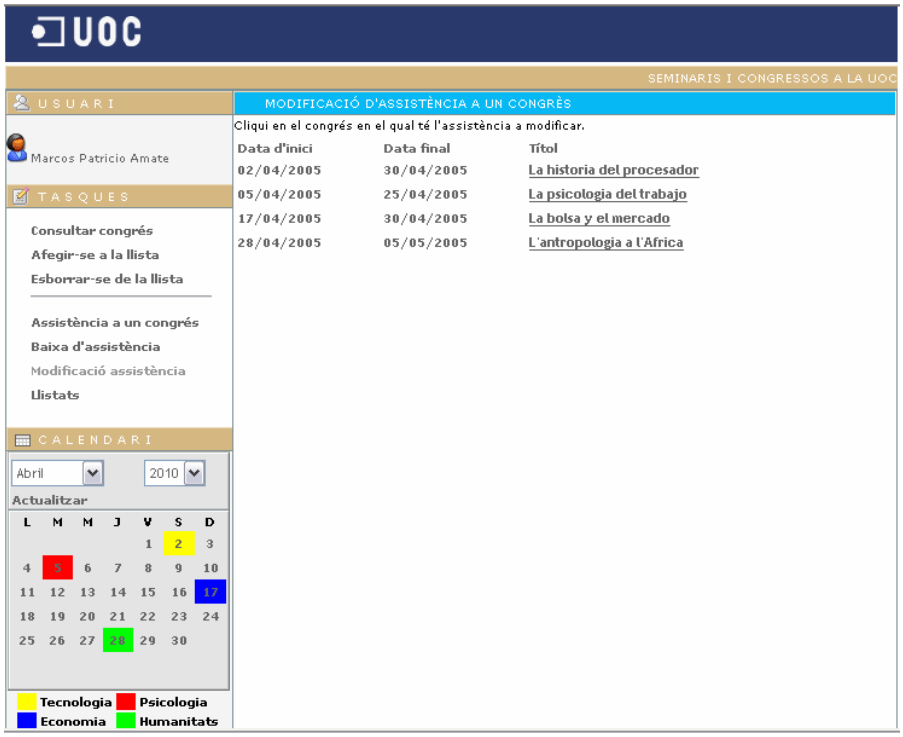
17) Clase BajaAsistenciaOk



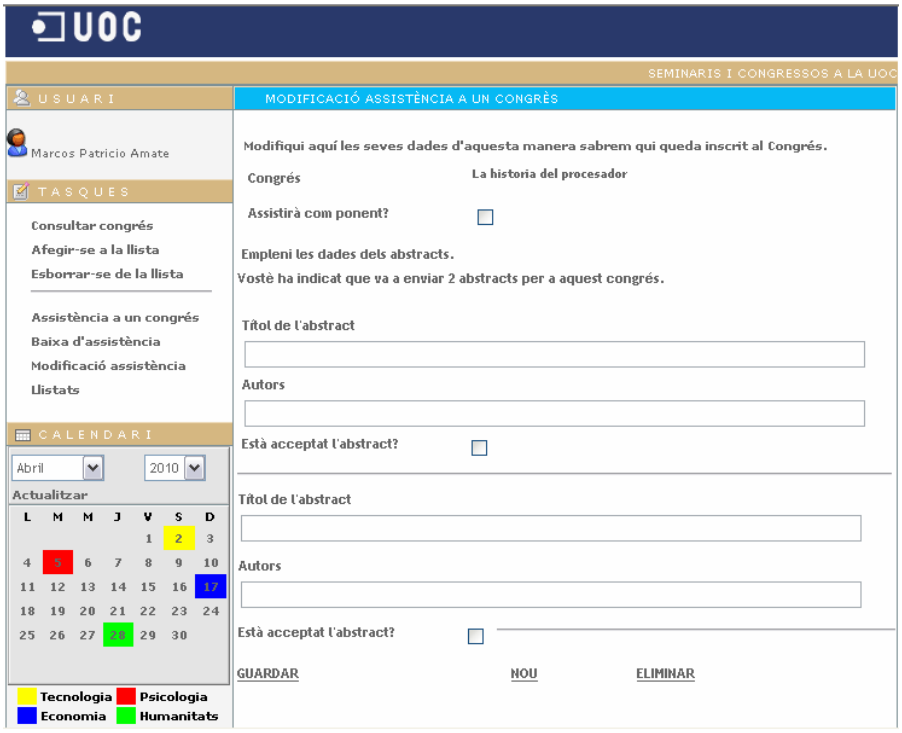
18) Clase ErrorBajaAsistencia



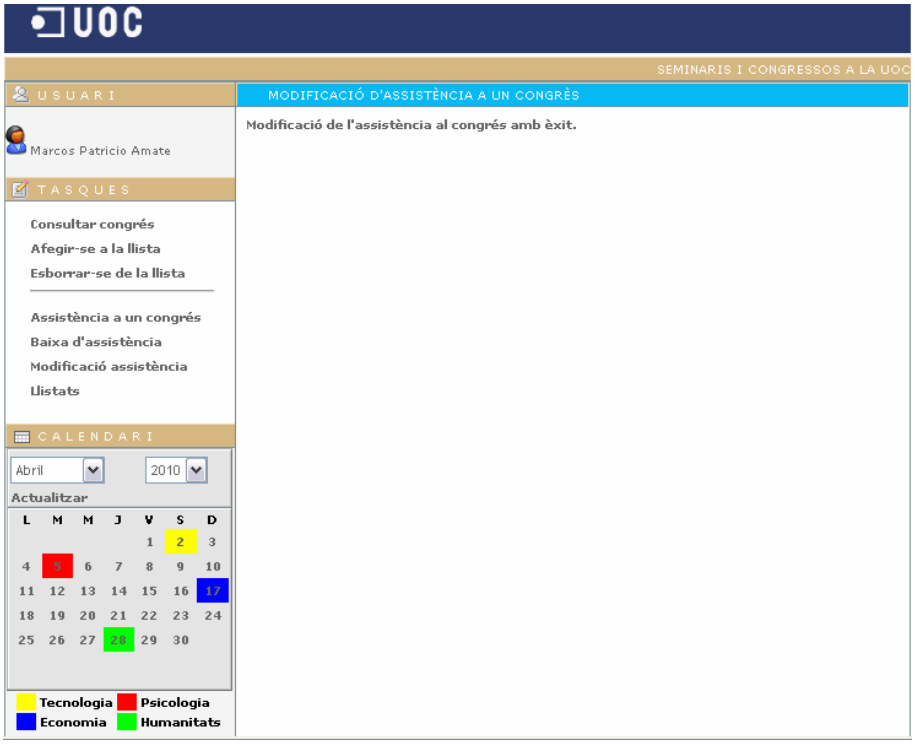
19) Clase ModificacióAssistència



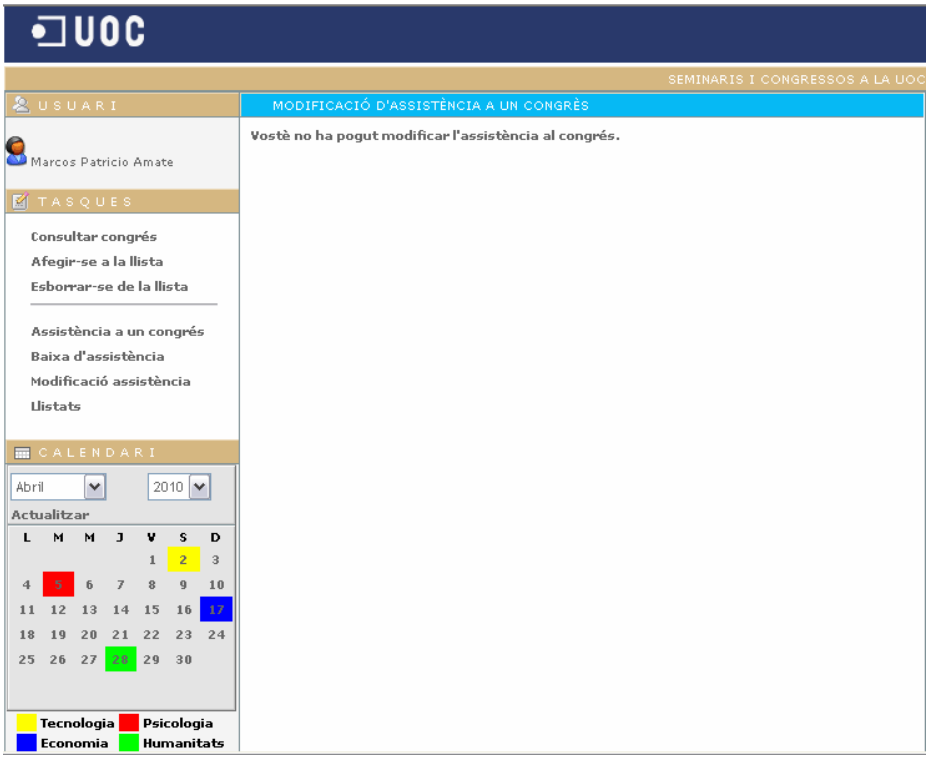
20) Clase ModificacióAssitència2



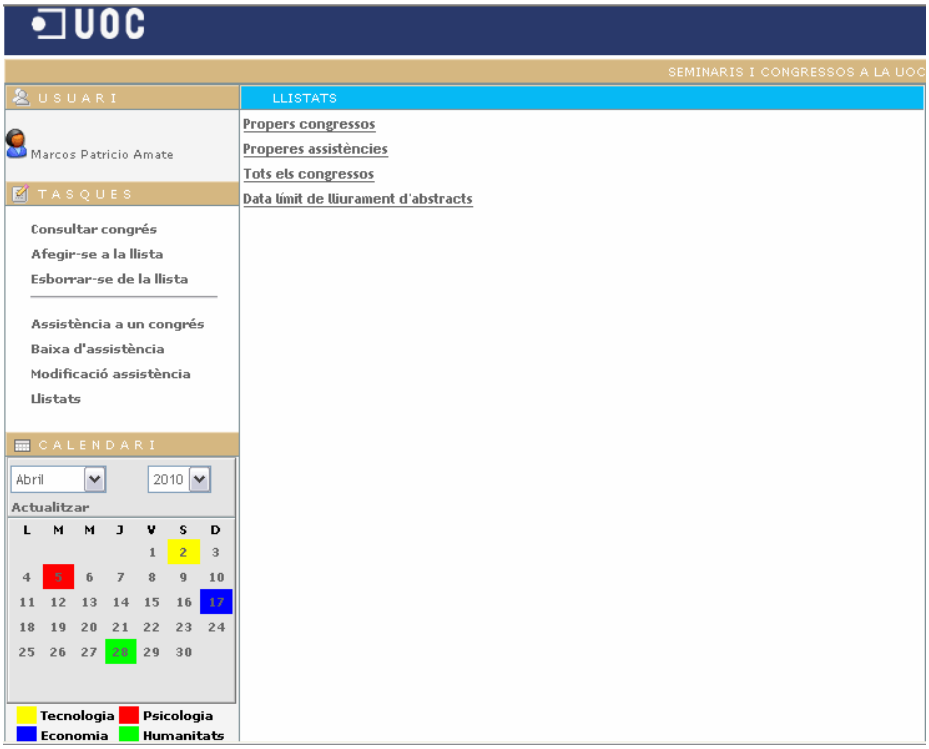
21) Clase ModificacióAssistènciaOK



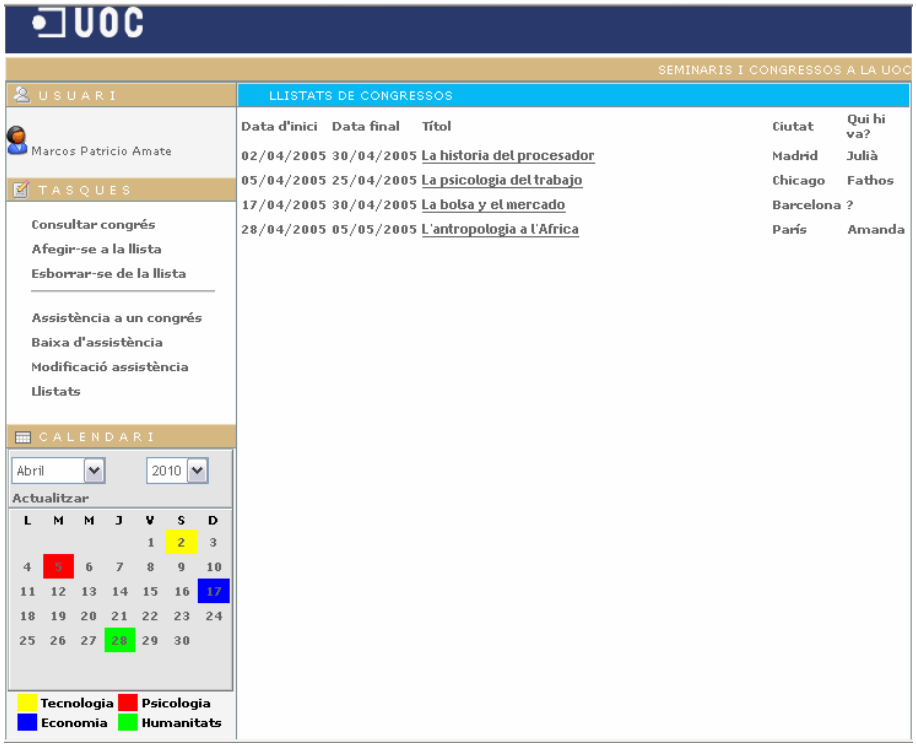
22) Clase ErrorModificacióAssistència



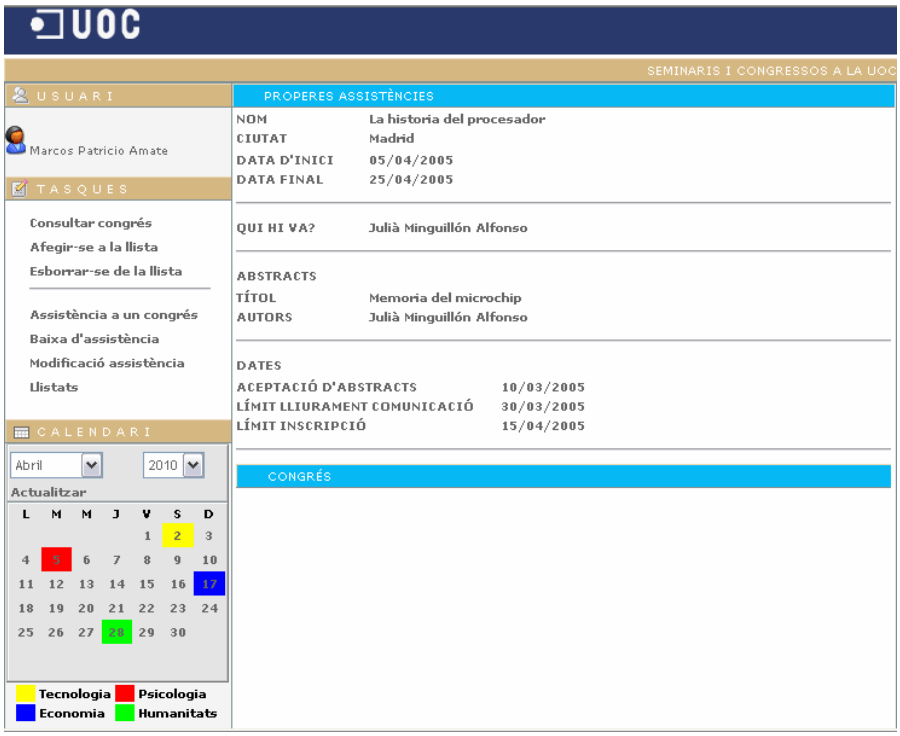
23) Clase Llistats



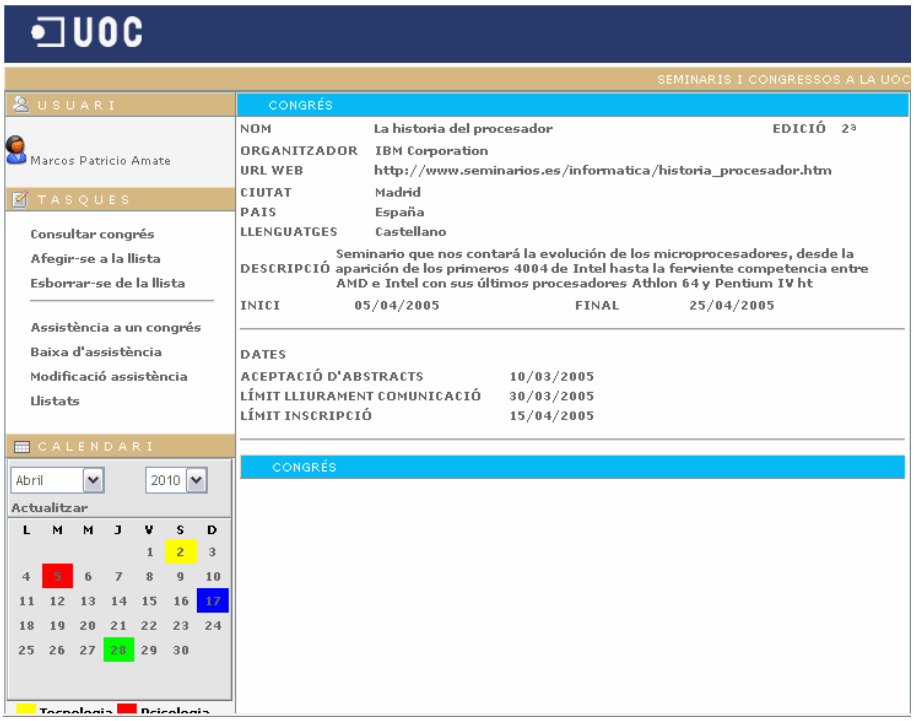
24) Clase LlistatsProximsCongressos



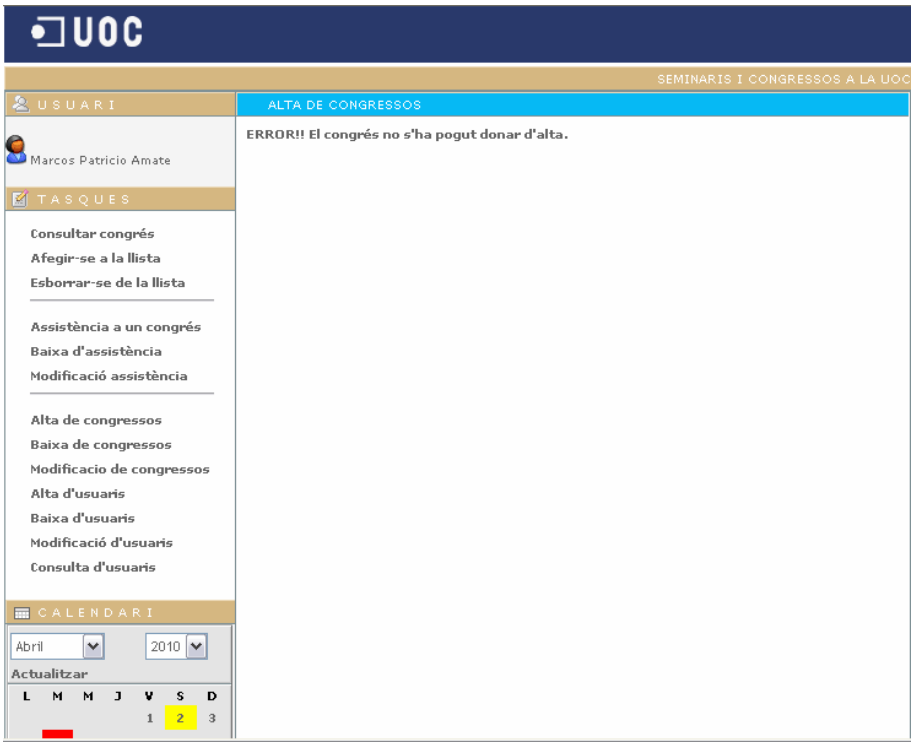
25) Clase LlistatsProximesAssistencies



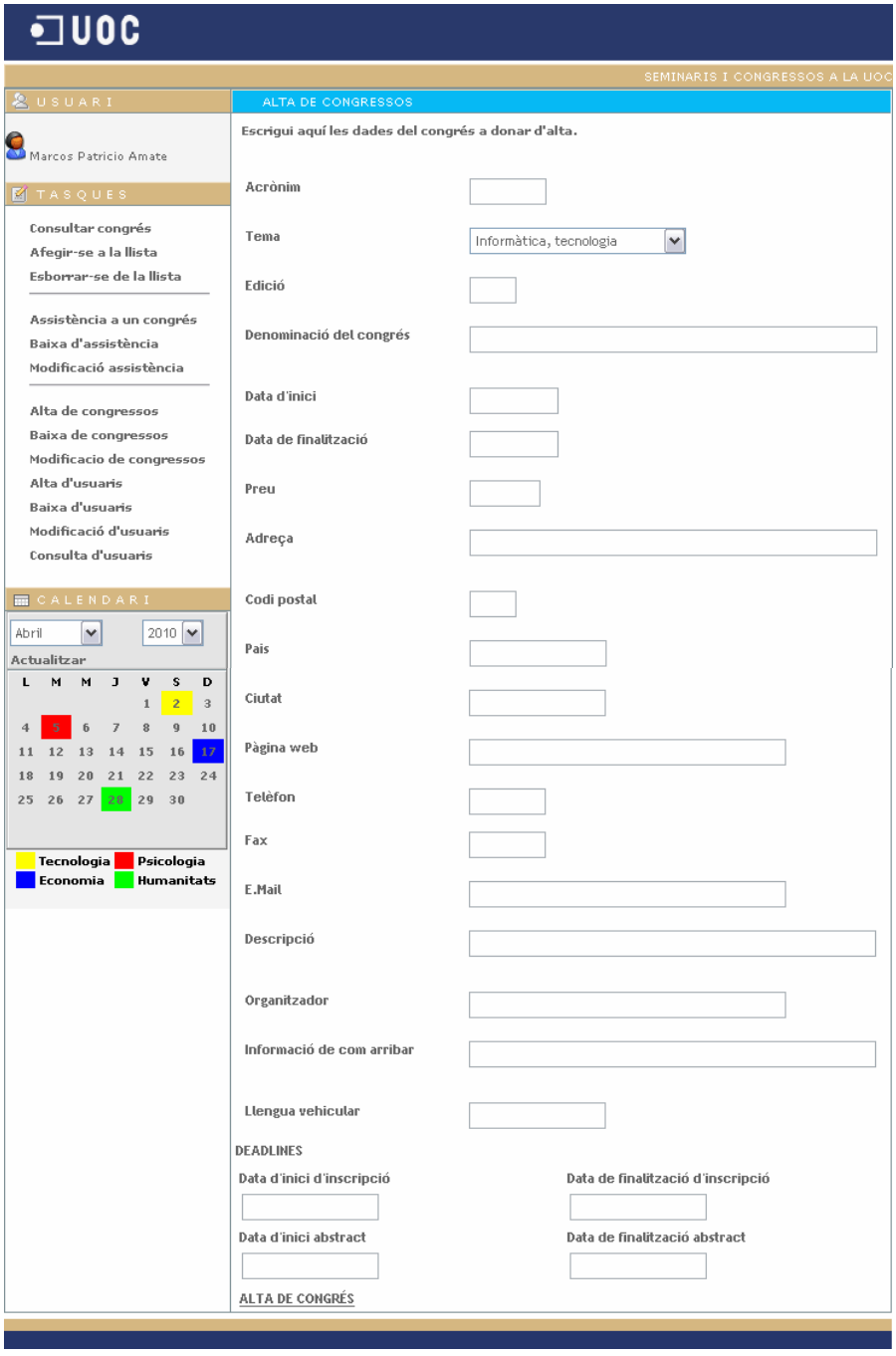
26) Clase PantallaTodosCongresos



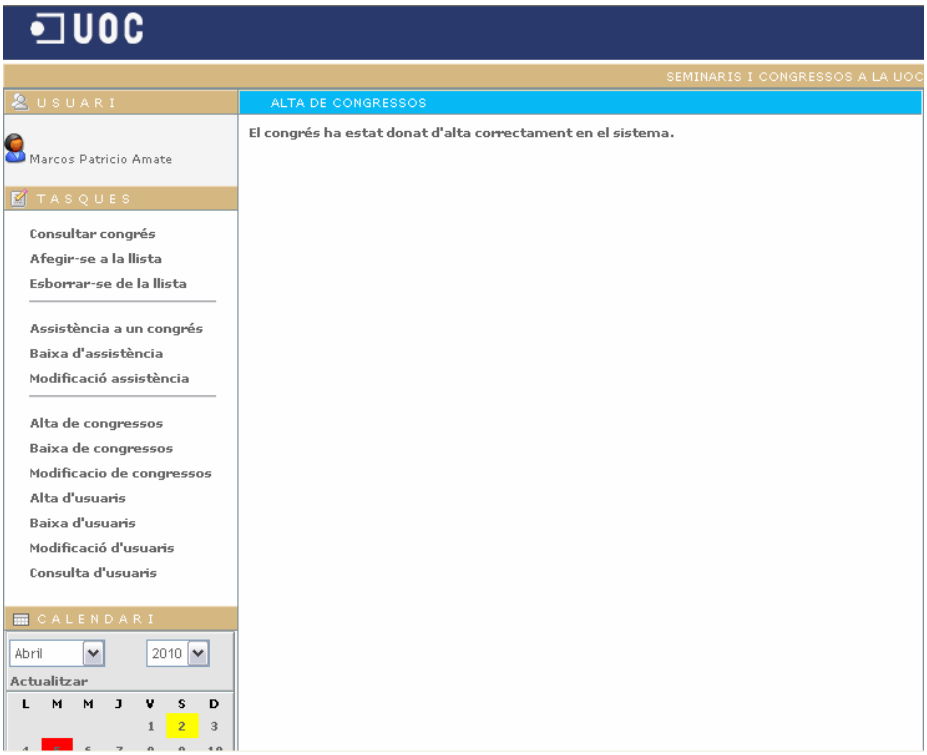
27) Clase PantallaErrorAltaCongresos



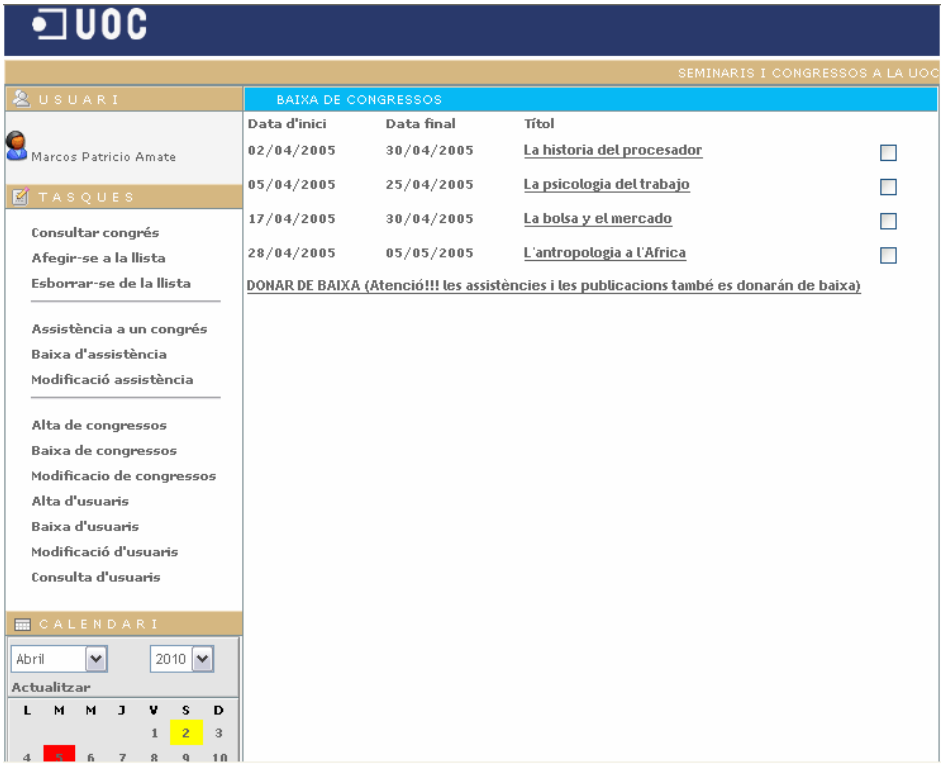
28) Clase PantallaAltaCongresos



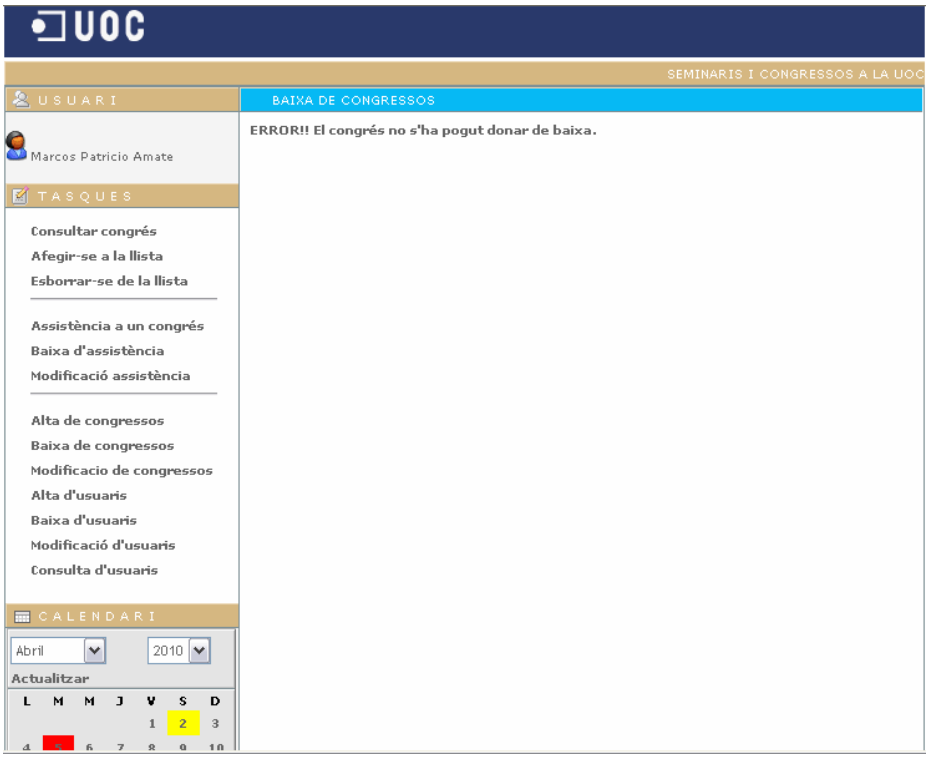
29) Clase PantallaAltaCongresosOk



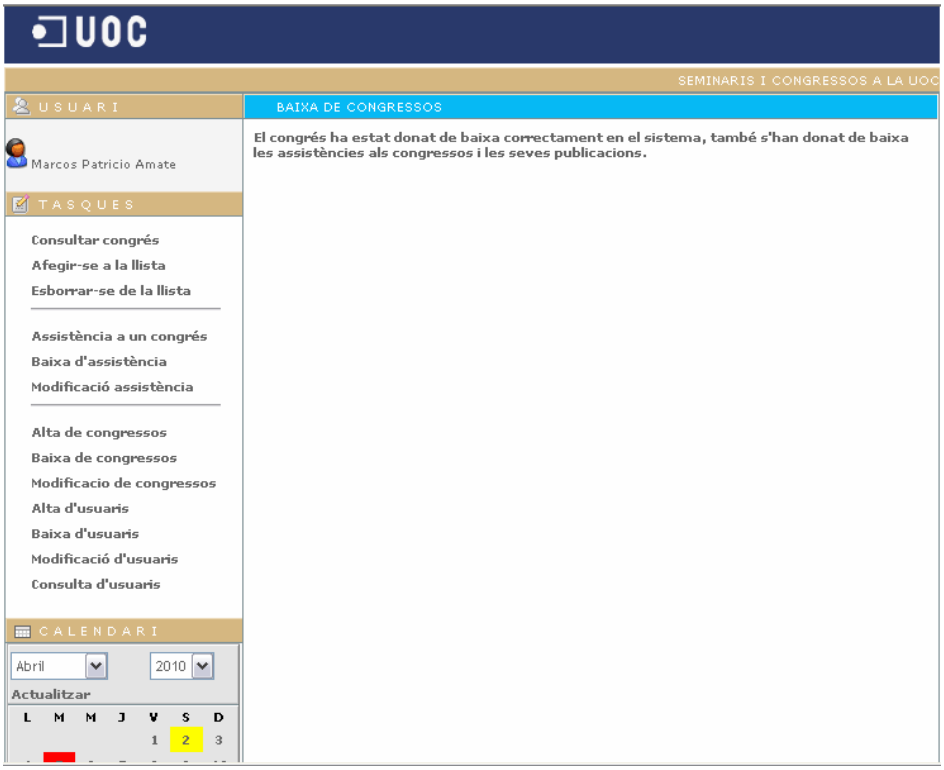
30) Clase PantallaBajaCongreso



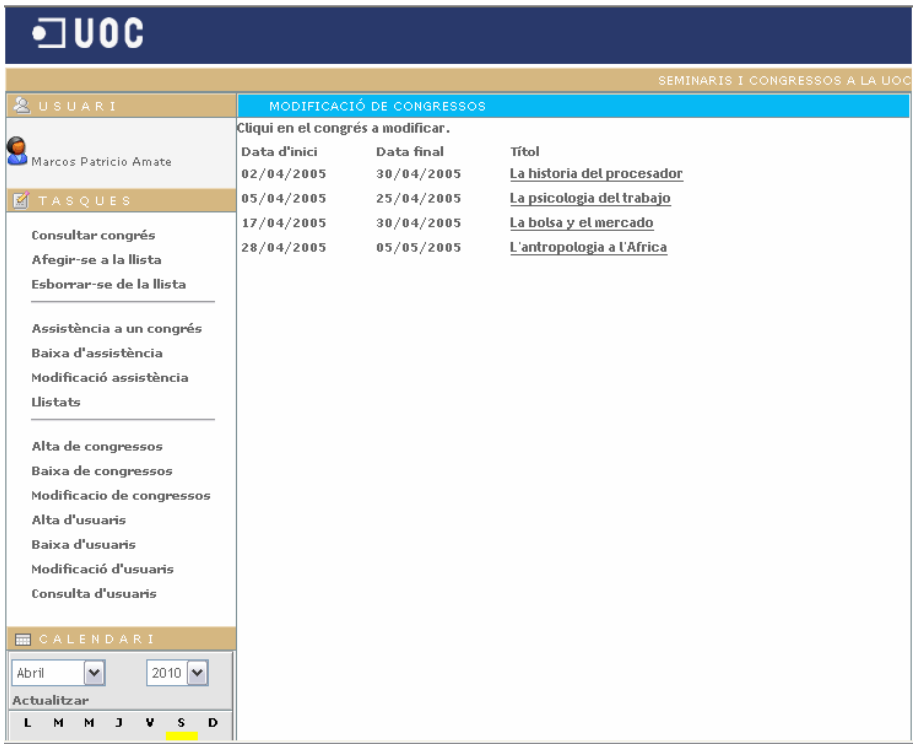
31) Clase ErrorBajaCongreso



32) Clase PantallaBajaCongresoOk

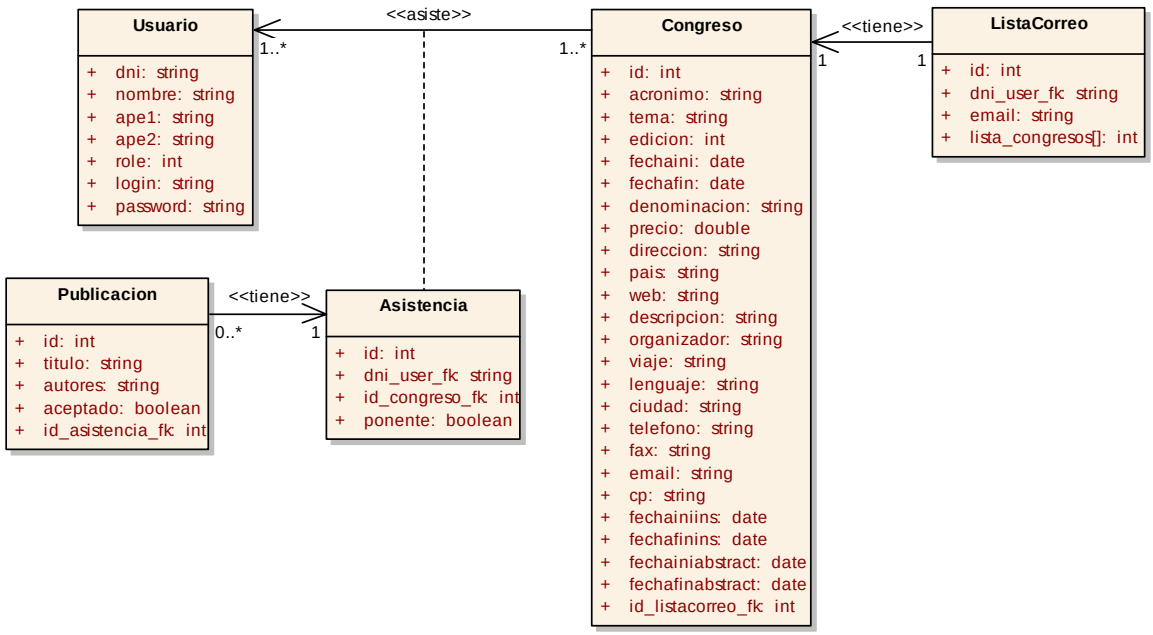


33) Clase PantallaModificaciónCongresos



El resto de pantallas sigue la misma tónica no pondré más pantallas ya que el documento se haría demasiado extenso.

3.10 Diagrama estático



4.- Diseño

4. 1.- El papel del diseño

La etapa de diseño hace de puente entre el análisis y la realización. Dentro del *Rational Unified Process*, el diseño se hace al final de la fase de elaboración y al comienzo de la de construcción.

Es una etapa obligatoriamente necesaria.

4.2.- La reutilización

La reutilización tiene cuatro modalidades:

1. De clases
2. De componentes
3. Los patrones
4. Los *frameworks*

Obviamente, la reutilización conlleva unas ventajas:

- Se reduce el desarrollo del programa
- Disminuye el trabajo de mantenimiento
- Mejora la fiabilidad
- El código reutilizado es mas eficiente
- Se gana en coherencia
- Preservación de la inversión

4.2.1.- Reutilización de clases

Básicamente se reutilizarán las clases que tienen funciones independientes del dominio. La reutilización de clases del mismo dominio sólo es posible en sistemas de software de la misma organización. Las clases que se reutilizan se suelen agrupar por funciones en librerías como los paquetes de Java.

4.2.2.- Reutilización de componentes

Un componente es un conjunto de clases, los objetos de las cuales colaboran en tiempo de ejecución con vista a llevar a cabo una función concreta.

Un componente se comportará como una clase y para reutilizarlo correctamente deberemos de conocer el contrato, que describe los efectos de las operaciones comprendidas en la interfaz y que invariantes tiene.

4.2.3.- Los patrones

Son una manera organizada de recoger la experiencia de los diseñadores de software para volverla a utilizar en casos semejantes.

Es una idea de diseño e implementación detallada y práctica, como una receta, que constituye un esbozo de solución de un problema que se presenta con una cierta frecuencia. Los detalles cambian para cada caso. El núcleo de un patrón es un par problema-solución.

Los patrones:

- Recogen experiencia
- Es algo bastante amplio
- Crean vocabulario
- Son un instrumento de documentación
- Si se utiliza el mismo patrón, facilitará la coherencia de los diseños para todo donde se presenta el mismo problema
- No dan una solución completa a un problema, sino genérica que es necesario completar.
- Ayudan a afrontar la complejidad del diseño.

4.2.4.- Los frameworks

El *framework,* es un conjunto de clases que constituye una aplicación incompleta y genérica. Si se completa, especializa, de manera adecuada se obtienen aplicaciones especializadas de un cierto tipo.

4.2.5.- Comparación entre patrones y frameworks

Los patrones son más pequeños, un *framework* puede contener diversos patrones, nunca al contrario
Los patrones son más abstractos, cada vez que se aplica un mismo patrón obtenemos un programa diferente.
Los patrones son menos especializados, se pueden utilizar en cualquier tipo de aplicación.

4.3.- El diseño arquitectónico

Tiene como objetivo definir las grandes líneas del modelo de diseño.

Actividades:

- Establecer la configuración de la red
- Decidir la utilización de un *framework* ya disponible

- Establecer los subsistemas

4.3.1.- Establecer la configuración de la red

Determinar los nodos que habrá, y las características que tendrán, conexiones entre estos y como serán, protocolos de comunicación en cuanto a ancho de banda, fiabilidad y calidad. En el caso del TFC, correrá dentro del campus de la UOC.

4.3.2.- Establecer los subsistemas

Los subsistemas pueden ser propios del proyecto, reutilizados de otros proyectos o bien programas del mercado, *middleware.*

Según la etapa previa de análisis tenemos los subsistemas siguientes:

Asistencias
Lista de correo
Consultas
Congresos

4.4.- Diseño de los casos de uso

Parte del diagrama de colaboración, y se consideran por separado las clases de frontera, de entidad y de control.

Las clases de frontera son la base de partida del diseño de la interficie de usuario. El diseño de las clases de entidad incluye el diseño de los instrumentos para gestionar la persistencia.

La implementación de la funcionalidad de los casos de uso se hace dentro de las clases de control y de las de entidad. Reglas:

Conviene que una de las clases de control dirija todo el proceso del caso de uso.
Es necesario aprovechar las oportunidades de reutilizar componentes y aplicar patrones. Para el diseño de las clases de control se estudia la implementación de las operaciones ya identificadas en el análisis una a una.

Mientras se diseñan los casos de uso se van incorporando clases y operaciones nuevas al diagrama estático del análisis, que se convertirá en el diagrama estático de diseño.

4.4.1.- Caso de uso número 1: "Consultar congreso"

El usuario puede consultar un congreso desde el calendario, haciendo un click en el día correspondiente a la fecha de inicio (marcada con un color), o bien puede realizar la consulta desde el menú. Ya sea por una vía u otra se llamará a la operación

consultar de la clase *Congreso*. Esta operación instancia la clase, abre una ventana y presenta el formato de pantalla PantallaBuscarCongreso, (si va por el calendario, se presentará directamente la PantallaDatosCongreso). Aquí podrá buscar por palabras clave o por identificador (si lo sabe). Si el usuario confirma, se llamará a la operación *consultar* de la clase *Congreso* con el identificador de congreso o las palabras clave. Esta operación puede no encontrar ningún congreso, dando un mensaje al usuario, una lista de congresos creando la PantallaListaCongreso, desde la cual podrá ir a un Congreso directamente o el congreso al cual buscaba instanciado la clase PantallaDatosCongreso.

4.4.2.- Caso de uso número 2: "Apuntarse a lista de correo"

Cuando el usuario selecciona la opción correspondiente en el menú, se llama a la operación crear de la clase *ListaCorreo*. Esta operación instancia la clase, determina el valor del número correlativo de la lista, abre una ventana e instancia el formato de PantallaApuntarseListaCorreo, además llamará a una operación proximoscongresos, pasándole como parámetro la fecha de hoy, que devolverá la lista para que el usuario pueda escogerlos.

En esta ventana el usuario introduce su DNI, y su email, y elige los congresos de los que quiere recibir información.

Si el usuario confirma se llama a la operación *apuntarse* de la clase *ListaCorreo* pasándole como parámetros el DNI, el email y el id de los congresos.

4.4.3.- Caso de uso número 3: "Darse de baja de la lista de correo"

Cuando el usuario selecciona la opción correspondiente en el menú, se llama a la operación *crear* de la clase *ListaCorreo*. Esta operación instancia la clase, abre una ventana e instancia el formato de PantallaBajaListaCorreo.

En esta ventana el usuario introduce su DNI, y si el usuario confirma se llama a la operación *borrarse* de la clase *ListaCorreo* pasándole como parámetros el DNI.

4.4.4.- Caso de uso número 4: "Apuntarse a un congreso"

Cuando el usuario selecciona la opción correspondiente en el menú, se llama a la operación crear de la clase *Asistencia*. Esta operación instancia la clase, determina el valor del número correlativo de la asistencia, abre una ventana e instancia el formato de PantallaApuntarseCongreso, además llamará a una operación proximoscongresos, pasándole como parámetro la fecha de hoy, que devolverá la lista para que el usuario pueda escoger el congreso al cual asistir. También llamará a la operación nabstracts de la clase *Asistencia* para mostrar los enviados y así poder modificarlos o si no tuviera ninguno se le habilitaría la posibilidad de introducir uno nuevo.

En esta ventana el usuario introduce si va a asistir como ponente y escogerá el congreso, podrá modificar los abstracts ya enviados con anterioridad o enviar uno nuevo.

Si el usuario confirma se llama a la operación *apuntarse* de la clase *Asistencia* pasándole como parámetros el DNI, el id de Congreso, y en caso de que haya enviado Publicaciones se llamará a la operación enviar de la clase *Publicacion* pasándole como parámetro del id de asistencia.

4.4.5.- Caso de uso número 5: "Darse de baja de un congreso"

Cuando el usuario selecciona la opción correspondiente en el menú, se llama a la operación buscar de la clase *Asistencia*. Esta operación buscará todos los congresos que el usuario esté inscrito, y abre una ventana e instancia el formato de PantallaBajaAsistencia.

Si el usuario confirma se llama a la operación *borrarse* de la clase *Asistencia* pasándole como parámetros los id de los congresos y el DNI, y en caso de que haya enviado Publicaciones se llamará a la operación borrar de la clase *Publicacion* pasándole como parámetro del id de asistencia.

4.4.6.- Caso de uso número 6: "Enviar documentación"

Esta opción la tiene que realizar desde la pantalla PantallaApuntarseCongreso, desde aquí aparecerán los abstracts enviados por el usuario, para poder modificarlos y también podrá enviar nuevos abstracts para ese congreso. Cuando el usuario pulsa en la opción enviar se llama a la operación crear de la clase *Publicacion*. Esta operación instancia la clase, determina el valor del número correlativo de la publicación, y en la misma ventana modificada, ahora se llamará PantallaEnvioPublicacion instancia el formato de PantallaEnvioPublicacion, además llamará a una operación nabstracts, pasándole como parámetro la el/los id de Asistencia para mostrar las n publicaciones que ya haya enviado el usuario para poderlas modificar.

En esta ventana el usuario introduce el título del abstract, los autores, y si está aceptado el abstract.

Si el usuario confirma se llama a la operación *enviar* de la clase *Publicacion* pasándole como parámetros el id de Asistencia.
Si el usuario pulsa sobre nueva publicación se llamará a la operación nuevo de la clase *Publicacion* que realizará un nuevo formulario para que el usuario pueda rellenar los datos del nuevo abstract. Y si pulsa sobre eliminar una vez seleccionados los abstracts, se llamará a la opción borrar de la clase *Publicacion* pasándole como parámetro el id de publicación.

4.4.7.- Caso de uso número 7: "Alta de congresos"

Cuando el administrador selecciona la opción correspondiente en el menú, se llama a la operación crear de la clase *Congreso*. Esta operación instancia la clase, determina el valor del número correlativo del Congreso, abre una ventana e instancia el formato de PantallaAltaCongreso.

En esta ventana el administrador introduce los datos relativos a un congreso. Si el usuario confirma se llama a la operación *alta* de la clase *Congreso* pasándole como parámetros todos los datos introducidos en el formulario por el administrador.

4.4.8.- Caso de uso número 8: "Baja de congresos"

Cuando el administrador selecciona la opción correspondiente en el menú, se llama a la operación buscar de la clase *Congreso*. Esta operación buscará todos los congresos que van a realizarse y están dados de alta en el sistema, y abre una ventana e instancia el formato de PantallaBajaCongreso, en la cual el administrador podrá seleccionar el/los congresos a dar de baja.

Si el usuario confirma se llama a la operación *borrar* de la clase *Congreso* pasándole como parámetros los id de los congresos, y en caso de que haya asistencias y publicaciones están también se borrarán, por lo que la operación borrar de la clase *Congreso* borrará el Congreso pertinente y en caso de que haya Asistencias, llamará a la operación borrar de *Asistencia* pasándole como parámetro el id de Congreso, y a su vez, si esta asistencia tuviese publicaciones también esta operación también llamará a la operación borrar de la clase *Publicacion*, pasándole como parámetro el id de asistencia.

4.4.9.- Caso de uso número 9: "Modificación de congresos"

Cuando el administrador selecciona la opción correspondiente en el menú, se llama a la operación modificar de la clase *Congreso*. Esta operación buscará todos los congresos que van a realizarse y están dados de alta en el sistema, y abre una ventana e instancia el formato de PantallaModificarCongreso, en la cual el administrador podrá seleccionar el/los congresos a modificar.

Si el usuario clica encima de un congreso se llamará a la operación buscar de la clase *Congreso* que devolverá todos los datos de ese congreso y se instanciará una clase PantallaModificaciónCongreso que estará rellenada con esos datos.

Si el usuario confirma se llama a la operación *modificar* de la clase *Congreso* pasándole como parámetros el id de los congresos.

4.4.10.- Caso de uso número 10: "Alta de usuarios"

Cuando el administrador selecciona la opción correspondiente en el menú, se llama a la operación crear de la clase *Usuario*. Esta operación instancia la clase, y abre una ventana e instancia el formato de PantallaAltaUsuario.

En esta ventana el administrador introduce los datos relativos a un usuario, asignándole una categoría de profesor o administrador del sistema, para el tema de privilegios. Si el usuario confirma se llama a la operación *alta* de la clase *Usuario* pasándole como parámetros todos los datos introducidos en el formulario por el administrador.

4.4.11.- Caso de uso número 11: "Baja de usuarios"

Cuando el administrador selecciona la opción correspondiente en el menú, se llama a la operación buscar de la clase *Usuario*. Esta operación buscará todos los usuarios que están dados de alta en el sistema, y abre una ventana e instancia el formato de PantallaBajaUsuario, en la cual el administrador podrá seleccionar el usuario a dar de baja.

Si el usuario confirma se llama a la operación *borrar* de la clase *Usuario* pasándole como parámetro el dni del usuario, y en caso de que haya asistencias y publicaciones están también se borrarán, por lo que la operación borrar de la clase *Usuario* borrará el Usuario pertinente y en caso de que haya Asistencias, llamará a la operación borrar de *Asistencia* pasándole como parámetro el dni del Usuario, y a su vez, si esta asistencia tuviese publicaciones también esta operación también llamará a la operación borrar de la clase *Publicacion*, pasándole como parámetro el id de asistencia. También se borrará la lista de correo del usuario pertinente, es decir se llamará a la operación borrar de la clase *ListaCorreo* pasándole como parámetro el dni del usuario.

4.4.12.- Caso de uso número 12: "Modificación de usuarios"

Cuando el administrador selecciona la opción correspondiente en el menú, se llama a la operación modificar de la clase *Usuario*. Esta operación buscará todos los usuarios que están dados de alta en el sistema, y abre una ventana e instancia el formato de PantallaModificarUsuario, en la cual el administrador podrá seleccionar el usuario a modificar.

Cuando el administrador escoge al usuario para modificar, se llamará a la operación buscar de la clase *Usuario* que devolverá todos los datos de ese usuario y se instanciará una clase PantallaModificaciónUsuario que estará rellenada con esos datos, no se permitirá modificar el DNI.

Si el usuario confirma se llama a la operación *modificar* de la clase *Usuario* pasándole como parámetros el dni del usuario.

4.4.13.- Caso de uso número 13: "Consulta de usuarios"

Cuando el administrador selecciona la opción correspondiente en el menú, se llama a la operación consultar de la clase *Usuario*. Esta operación buscará todos los usuarios que están dados de alta en el sistema, y abre una ventana e instancia el formato de PantallaConsultaDeUsuarios, en la cual el administrador podrá seleccionar el usuario a consultar.

Cuando el administrador escoge al usuario para consultar, se llamará a la operación buscar de la clase *Usuario* que devolverá todos los datos de ese usuario y se instanciará una clase PantallaConsultarUsuario que estará rellenada con esos datos.

4.4.14.- Caso de uso número 14: "Login"

Cuando el usuario se identifica en la aplicación, se llama a la operación crear de la clase *Usuario*, a continuación se llama internamente a la operación buscar de la clase *Usuario* pasándole como parámetros el login y password, para verificar si existe algún usuario con ese login y ese password, si es cierto se abrirá una ventana de bienvenida al sistema como usuario identificado en el sistema y tendrá las opciones pertinentes de acuerdo a su categoría (profesor o administrador) y abre una ventana e instancia el formato de PantallaBienvenidaLogin.

Si nos devolviese un error, el usuario no existe, instanciaría la clase PantallaErrorLogin, indicando que el usuario no existe.

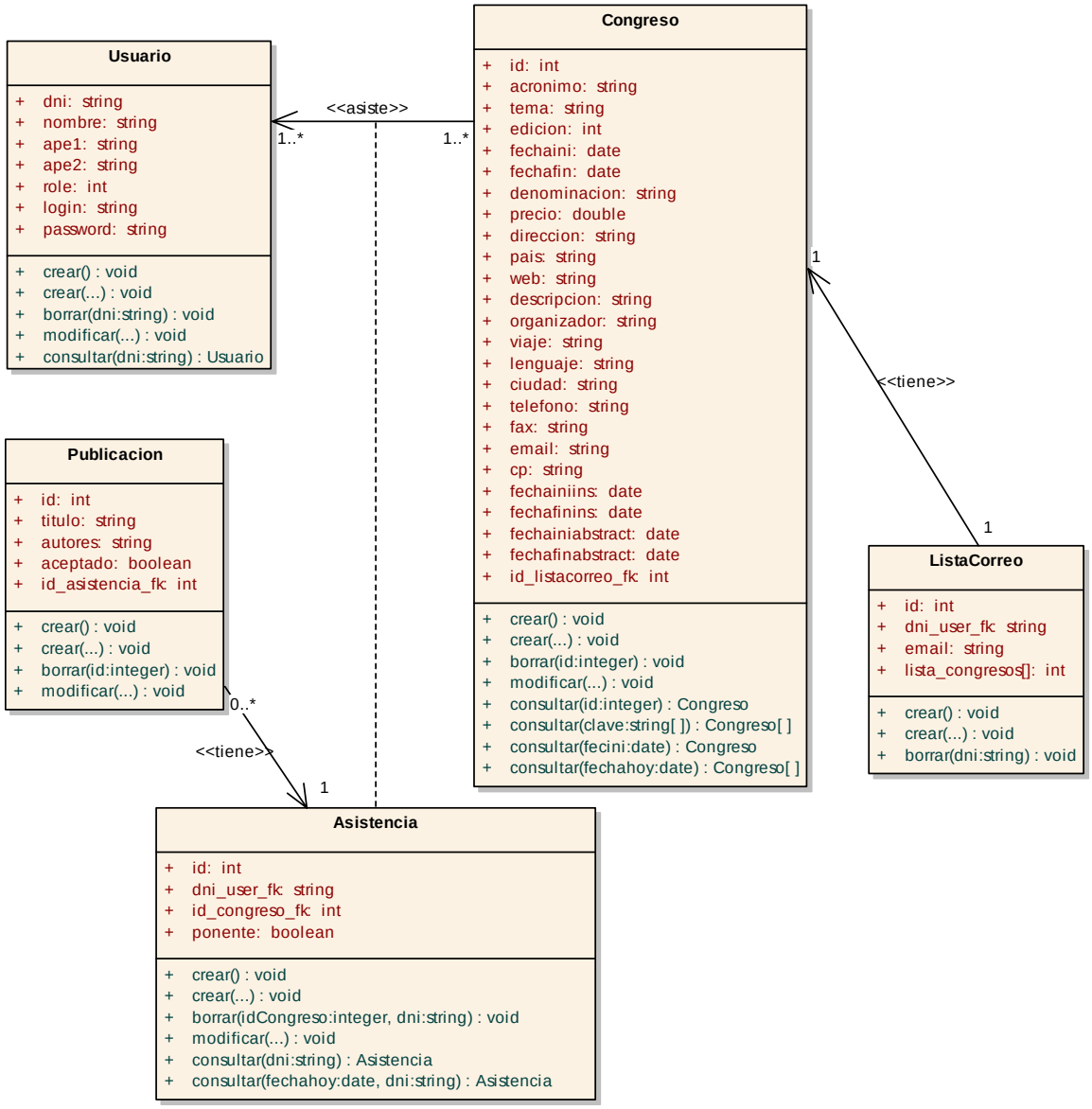
4.5.- Revisión diagrama estático de diseño

La revisión del diagrama estático de diseño, tiene en cuenta los siguientes aspectos: la normalización de los nombres, reutilización de clases, la adaptación de la herencia al nivel soportado por el lenguaje de programación, la mejora del rendimiento, el incremento de la velocidad y la reducción del tránsito de mensajes, clases temporales para almacenar datos intermedios.

En las clases del diagrama estático de diseño se indican solamente aquellos atributos que se han añadido en la versión de análisis para implementar las asociaciones, las operaciones se indican todas.

No tengo ninguna herencia que eliminar, puesto que no hay ninguna en el diagrama estático.

El diagrama estático queda finalmente:



4.6.- Diseño de la persistencia

Para la persistencia del proyecto utilizaré una base de datos relacional, concretamente MySQL.

Para la elaboración del diseño de la persistencia, primeramente debemos eliminar la herencia, pero como ha quedado claro en el punto anterior, en este caso no es necesario ya que no la he utilizado.

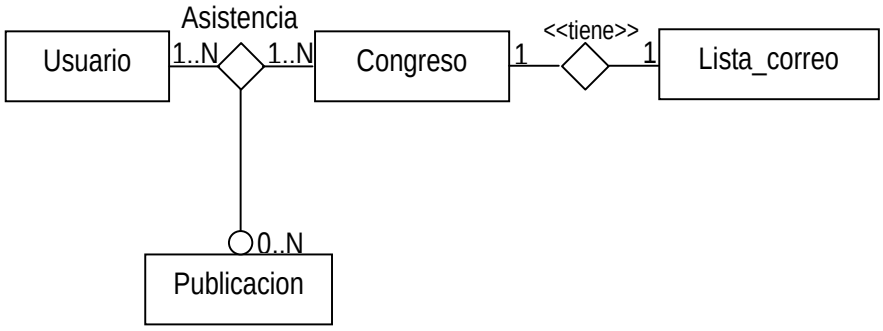
A continuación, tengo que elaborar el diagrama de Entidad – Relación equivalente al diagrama estático de las clases persistentes y finalmente obtendremos la especificación de la base de datos relacional.

Para efectuar este proceso determino que clases deben de ser persistentes:

Clases persistentes: *Usuario, Congreso, Lista_correo, Asistencia, Publicacion.*

Clases temporales: *Role, Tema, Viaje.*

El diagrama Entidad – Relación que corresponde a este proyecto es el siguiente:



4.7.- Base de datos relacional

Tablas de la base de datos:

Tabla usuario	Tipo	Claves
dni	String	Clave Primaria
nombre	String	
ape1	String	
ape2	String	
role	Integer	
login	String	
password	String	

Tabla congreso	Tipo	Claves
id	Integer	Clave Primaria
acronimo	String	
tema	String	
edicion	Integer	
fechaini	Date	
fechafin	Date	
denominacion	String	

precio	Double	
direccion	String	
pais	String	
web	String	
descripcion	String	
organizador	String	
viaje	String	
lenguaje	String	
ciudad	String	
telefono	String	
fax	String	
email	String	
cp	String	
fechainiins	Date	
fechafinins	Date	
fechainiabstract	Date	
fechafinabstract	Date	
idlistacorreofk	Integer	Llave Extranjera en listacorreo

Tabla listacorreo	**Tipo**	**Claves**
id	Integer	Clave Primaria
email	String	
listacongresos	Integer[]	
dniuserfk	String	Llave Extranjera en usuario

Tabla asistencia	**Tipo**	**Claves**
id	Integer	Clave Primaria
ponente	Integer	
idcongresofk	Integer	Llave Extranjera en congreso
dniuserfk	String	Llave Extranjera en usuario

Tabla publicacion	**Tipo**	**Claves**
id	Integer	Clave Primaria
titulo	String	
autores	String	
aceptado	Boolean	
idasistenciafk	Integer	Llave Extranjera en asistencia

4.8.- Diseño de la interfaz de usuario

El diseño de la interfaz de usuario sirve para ver como se van a representar los diferentes datos, como se implementan los diálogos y el diseño de algunas ventanas.

Los datos se presentarán de la misma manera en todos los formularios, y para ver la representación de cómo se hará la presentación ver el apartado análisis de la interfaz de usuario de este proyecto donde ya están impresas las pantallas de la aplicación.

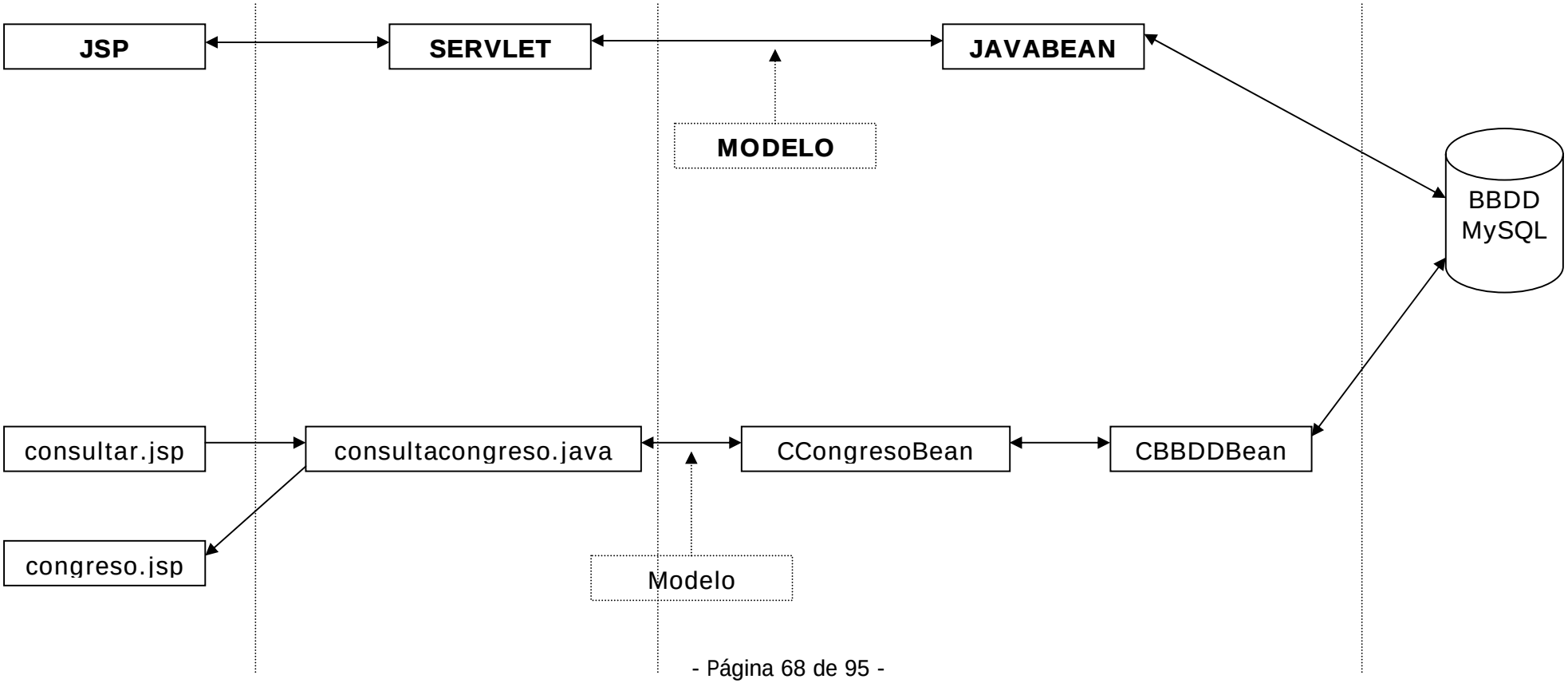
5.- Implementación

5.1. Java

Para la implementación del código se ha utilizado el lenguaje de programación Java, que permite su inclusión en el navegador mediante tecnología JSP. Así se garantiza la portabilidad entre diferentes plataformas, sistemas operativos y navegadores.

Así he utilizado la arquitectura de 3 capas basada en J2EE.

Se pretende separar claramente la presentación, la lógica de negocio y el acceso a datos. Concretamente, la presentación se prevé implementar con combinación de servlets y JSPs, la lógica de negocio y la capa de datos con Java Beans, tal y como muestra el siguiente esquema con un ejemplo de uso:



5.2.- Ejemplo de codificación del caso de uso consulta de un congreso

CAPA JSP
Consultar.jsp

```
<html>
   <head>
      <title>Consulta de seminaris i congressos a la UOC.</title>
      <meta http-equiv="content-type" content="text/html;charset=utf-8"></meta>
      <link rel="stylesheet" href="./css/agenda.css" type="text/css"></link>
      <script src="/js/funciones.js"></script>
   </head>
   <body>
      <div id="pagewidth">
         <%@ include file="modulos/cabecera.jsp"%>
         <div id="outer">
            <div id="inner">
               <div id="leftcol">
                  <%@ include file="modulos/usuario.jsp"%>
                  <%@ include file="modulos/menu.jsp"%>
                  <%@ include file="modulos/calendari.jsp"%>
               </div>
               <div id="maincol">
                  <h2>CONSULTA DE CONGRESSOS</h2>
                     <form name="formCONSULTA" method="POST"
                     action="/servlet/consultacongreso">
                     <input type="hidden" name="field"
                     value="<%=request.getParameter("field")%>">
                     <table width="100%" border="0" cellspacing="5"
                     cellpadding="3">
                        <tr>
                         <td width="100%">
                            Si sap l'acrónim del Congrés introdueix-ho, sinó introdueixi
                              les paraules claus a cercar al camp corresponent.
                         </td>
                        </tr>
                        <tr>
                            <td width="100%" align="left">
                            <label>Acrónim del Congrès</label>
                            <input size="10" type="text"
                             name="idCongreso"></input>

                            </td>
                        </tr>
```

```
                        <br>

                        <tr>
                            <td align="left" class="texto9">
                            <label>Congressos amb les paraules clau</label>
                            <input size="10" type="text" name="clave"></input>
                            <input type="button" value="Cercar"
                             onClick="Validar(this.form)"></input>

                            </td>
                        </tr>
                    </table>
                    </form>
                    <div class="clr"></div>
                </div>
            </div>
            <div class="clr"></div>
        </div>
        <%@ include file="modulos/footer.jsp"%>
        <div class="clr">
        </div>
    </div>
  </body>
</html>
```

CAPA SERVLET

consultacongreso.java

```
package servlet;

import javabean.CCongresoBean;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.Collection;

/**
 * Creado por Marcos Patricio Amate
 * User: Marcos
 * Date: 26-abr-2005
 * Time: 17:49:11
 */
```

```
public class consultacongreso extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse
        response)throws IOException,ServletException,NullPointerException
    {
      try
      {
         CCongresoBean congreso = new CCongresoBean();
         String idCongreso = null;
         String clave = null;
         idCongreso = request.getParameter("idCongreso");
         clave = request.getParameter("clave");
         Collection c = congreso.consultacongreso(idCongreso,clave);
         if(c.isEmpty())
         {
             String destino="/errorbuscacongreso.jsp";
             RequestDispatcher rd =
               getServletContext().getRequestDispatcher(destino);
             rd.forward(request, response);
         }else
         {

       request.setAttribute("congreso",congreso.consultacongreso(idCongreso,clave))
       ;
             String destino="/congreso.jsp";
             RequestDispatcher rd =
               getServletContext().getRequestDispatcher(destino);
             rd.forward(request, response);
         }
      }catch(NullPointerException e)
      {
         String destino="/errorbuscacongreso.jsp";
         RequestDispatcher rd = getServletContext().getRequestDispatcher(destino);
         rd.forward(request, response);
      }

    }
    public void doPost(HttpServletRequest request, HttpServletResponse
        response)throws IOException,ServletException
    {

       doGet(request,response);
    }
}
```

CAPA JAVABEAN

CcongresoBean.java

```
package javabean;

import modelos.CCongresoModelo;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Collection;
import java.util.Date;
import java.util.Vector;

/**
 * Creado por Marcos Patricio Amate
 * User: Marcos
 * Date: 28-abr-2005
 * Time: 8:59:28
 */
public class CCongresoBean
{
    private CBBDDBean bd;
    private String sqlstr = null;
    private int y = 0;
    public CCongresoBean()
    {
        bd = new CBBDDBean();
        bd.setServidor("localhost");
        bd.setNombre("congresos");
        bd.setPuerto("8080");
    }
    public Collection consultacongresos(Date hoy)
    {
        Collection congresos = new Vector();
        String ahora = new java.text.SimpleDateFormat("yyyy/MM/dd").format(hoy);
        sqlstr = "SELECT * FROM congreso WHERE fechaini >= '" + ahora + "' ORDER
BY(fechaini)";
        bd.setConsulta(sqlstr);
        ResultSet rs = bd.getResultado();
        try
        {
            while (rs.next())
            {
```

```
            CCongresoModelo congresomodelo = new CCongresoModelo();
            congresomodelo.setAcronimo(rs.getString("acronimo"));
            congresomodelo.setTema(rs.getString("tema"));
            congresomodelo.setEdicion(rs.getInt("edicion"));
            congresomodelo.setInicio(rs.getDate("fechaini"));
            congresomodelo.setFinalizacion(rs.getDate("fechafin"));
            congresomodelo.setDenominacion(rs.getString("denominacion"));
            congresomodelo.setPrecio(rs.getDouble("precio"));
            congresomodelo.setDireccion(rs.getString("direccion"));
            congresomodelo.setPais(rs.getString("pais"));
            congresomodelo.setUrl(rs.getString("web"));
            congresomodelo.setDescripcion(rs.getString("descripcion"));
            congresomodelo.setOrganizador(rs.getString("organizador"));
            congresomodelo.setViaje(rs.getString("viaje"));
            congresomodelo.setLenguaje(rs.getString("lenguaje"));
            congresomodelo.setCiudad(rs.getString("ciudad"));
            congresomodelo.setTelefono(rs.getString("telefono"));
            congresomodelo.setFaximil(rs.getString("fax"));
            congresomodelo.setEmail(rs.getString("email"));
            congresomodelo.setCodigo(rs.getString("cp"));
            congresomodelo.setInicioins(rs.getDate("fechainiins"));
            congresomodelo.setFinalizacionins(rs.getDate("fechafinins"));
            congresomodelo.setInicioabs(rs.getDate("fechainiabstract"));
            congresomodelo.setFinalizacionabs(rs.getDate("fechafinabstract"));
            congresomodelo.setIdlistacorreo(rs.getInt("idlistacorreofk"));
            congresos.add(congresomodelo);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

            return congresos;
    }

    public Collection consultacongreso(String idCongreso,String clave)
      {
        Collection congresos = new Vector();
        sqlstr = "SELECT * FROM congreso WHERE 1 = 1";
        if(!idCongreso.equals(""))
        {
            sqlstr += " AND acronimo = '" + idCongreso + "'";
        }
        if(!clave.equals(""))
        {
            sqlstr += " AND denominacion = '" + clave + "'";
        }
        if(clave.equals("") && (idCongreso.equals("")))
```

```
        {
            return null;
        }
        bd.setConsulta(sqlstr);
        ResultSet rs = bd.getResultado();
        try
        {
            while (rs.next())
            {
            CCongresoModelo congresomodelo = new CCongresoModelo();
            congresomodelo.setAcronimo(rs.getString("acronimo"));
            congresomodelo.setTema(rs.getString("tema"));
            congresomodelo.setEdicion(rs.getInt("edicion"));
            congresomodelo.setInicio(rs.getDate("fechaini"));
            congresomodelo.setFinalizacion(rs.getDate("fechafin"));
            congresomodelo.setDenominacion(rs.getString("denominacion"));
            congresomodelo.setPrecio(rs.getDouble("precio"));
            congresomodelo.setDireccion(rs.getString("direccion"));
            congresomodelo.setPais(rs.getString("pais"));
            congresomodelo.setUrl(rs.getString("web"));
            congresomodelo.setDescripcion(rs.getString("descripcion"));
            congresomodelo.setOrganizador(rs.getString("organizador"));
            congresomodelo.setViaje(rs.getString("viaje"));
            congresomodelo.setLenguaje(rs.getString("lenguaje"));
            congresomodelo.setCiudad(rs.getString("ciudad"));
            congresomodelo.setTelefono(rs.getString("telefono"));
            congresomodelo.setFaximil(rs.getString("fax"));
            congresomodelo.setEmail(rs.getString("email"));
            congresomodelo.setCodigo(rs.getString("cp"));
            congresomodelo.setInicioins(rs.getDate("fechainiins"));
            congresomodelo.setFinalizacionins(rs.getDate("fechafinins"));
            congresomodelo.setInicioabs(rs.getDate("fechainiabstract"));
            congresomodelo.setFinalizacionabs(rs.getDate("fechafinabstract"));
            congresomodelo.setIdlistacorreo(rs.getInt("idlistacorreofk"));
            congresos.add(congresomodelo);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

            return congresos;
    }
  public Collection consultacongresobaja(String acronimo,String dni)
    {
        Collection congresos = new Vector();
        sqlstr = "SELECT * FROM congreso WHERE acronimo = '" + acronimo + "'";
        bd.setConsulta(sqlstr);
```

```
        ResultSet rs = bd.getResultado();
        try
        {
           while (rs.next())
           {
           CCongresoModelo congresomodelo = new CCongresoModelo();
           congresomodelo.setAcronimo(rs.getString("acronimo"));
           congresomodelo.setTema(rs.getString("tema"));
           congresomodelo.setEdicion(rs.getInt("edicion"));
           congresomodelo.setInicio(rs.getDate("fechaini"));
           congresomodelo.setFinalizacion(rs.getDate("fechafin"));
           congresomodelo.setDenominacion(rs.getString("denominacion"));
           congresomodelo.setPrecio(rs.getDouble("precio"));
           congresomodelo.setDireccion(rs.getString("direccion"));
           congresomodelo.setPais(rs.getString("pais"));
           congresomodelo.setUrl(rs.getString("web"));
           congresomodelo.setDescripcion(rs.getString("descripcion"));
           congresomodelo.setOrganizador(rs.getString("organizador"));
           congresomodelo.setViaje(rs.getString("viaje"));
           congresomodelo.setLenguaje(rs.getString("lenguaje"));
           congresomodelo.setCiudad(rs.getString("ciudad"));
           congresomodelo.setTelefono(rs.getString("telefono"));
           congresomodelo.setFaximil(rs.getString("fax"));
           congresomodelo.setEmail(rs.getString("email"));
           congresomodelo.setCodigo(rs.getString("cp"));
           congresomodelo.setInicioins(rs.getDate("fechainiins"));
           congresomodelo.setFinalizacionins(rs.getDate("fechafinins"));
           congresomodelo.setInicioabs(rs.getDate("fechainiabstract"));
           congresomodelo.setFinalizacionabs(rs.getDate("fechafinabstract"));
           congresomodelo.setIdlistacorreo(rs.getInt("idlistacorreofk"));
           congresomodelo.setDni(dni);
           congresos.add(congresomodelo);
           }
        } catch (SQLException e) {
           e.printStackTrace();
        }

           return congresos;
     }

   public Collection consultacongresacr(String acronimo)
        {
           Collection congresos = new Vector();
           sqlstr = "SELECT * FROM congreso WHERE acronimo = '" + acronimo +
"'";
           bd.setConsulta(sqlstr);
           ResultSet rs = bd.getResultado();
```

```
            try
            {
               while (rs.next())
               {
               CCongresoModelo congresomodelo = new CCongresoModelo();
               congresomodelo.setAcronimo(rs.getString("acronimo"));
               congresomodelo.setTema(rs.getString("tema"));
               congresomodelo.setEdicion(rs.getInt("edicion"));
               congresomodelo.setInicio(rs.getDate("fechaini"));
               congresomodelo.setFinalizacion(rs.getDate("fechafin"));
               congresomodelo.setDenominacion(rs.getString("denominacion"));
               congresomodelo.setPrecio(rs.getDouble("precio"));
               congresomodelo.setDireccion(rs.getString("direccion"));
               congresomodelo.setPais(rs.getString("pais"));
               congresomodelo.setUrl(rs.getString("web"));
               congresomodelo.setDescripcion(rs.getString("descripcion"));
               congresomodelo.setOrganizador(rs.getString("organizador"));
               congresomodelo.setViaje(rs.getString("viaje"));
               congresomodelo.setLenguaje(rs.getString("lenguaje"));
               congresomodelo.setCiudad(rs.getString("ciudad"));
               congresomodelo.setTelefono(rs.getString("telefono"));
               congresomodelo.setFaximil(rs.getString("fax"));
               congresomodelo.setEmail(rs.getString("email"));
               congresomodelo.setCodigo(rs.getString("cp"));
               congresomodelo.setInicioins(rs.getDate("fechainiins"));
               congresomodelo.setFinalizacionins(rs.getDate("fechafinins"));
               congresomodelo.setInicioabs(rs.getDate("fechainiabstract"));
               congresomodelo.setFinalizacionabs(rs.getDate("fechafinabstract"));
               congresomodelo.setIdlistacorreo(rs.getInt("idlistacorreofk"));
               congresos.add(congresomodelo);
               }
            } catch (SQLException e) {
               e.printStackTrace();
            }

               return congresos;
         }
   public CCongresoModelo consultacongresacronimo(String acronimo)
            {
               CCongresoModelo congresomodelo = new CCongresoModelo();
               sqlstr = "SELECT * FROM congreso WHERE acronimo = '" + acronimo +
"'";
               bd.setConsulta(sqlstr);
               ResultSet rs = bd.getResultado();
               try
               {
                  while (rs.next())
```

```
                {
                congresomodelo.setAcronimo(rs.getString("acronimo"));
                congresomodelo.setTema(rs.getString("tema"));
                congresomodelo.setEdicion(rs.getInt("edicion"));
                congresomodelo.setInicio(rs.getDate("fechaini"));
                congresomodelo.setFinalizacion(rs.getDate("fechafin"));
                congresomodelo.setDenominacion(rs.getString("denominacion"));
                congresomodelo.setPrecio(rs.getDouble("precio"));
                congresomodelo.setDireccion(rs.getString("direccion"));
                congresomodelo.setPais(rs.getString("pais"));
                congresomodelo.setUrl(rs.getString("web"));
                congresomodelo.setDescripcion(rs.getString("descripcion"));
                congresomodelo.setOrganizador(rs.getString("organizador"));
                congresomodelo.setViaje(rs.getString("viaje"));
                congresomodelo.setLenguaje(rs.getString("lenguaje"));
                congresomodelo.setCiudad(rs.getString("ciudad"));
                congresomodelo.setTelefono(rs.getString("telefono"));
                congresomodelo.setFaximil(rs.getString("fax"));
                congresomodelo.setEmail(rs.getString("email"));
                congresomodelo.setCodigo(rs.getString("cp"));
                congresomodelo.setInicioins(rs.getDate("fechainiins"));
                congresomodelo.setFinalizacionins(rs.getDate("fechafinins"));
                congresomodelo.setInicioabs(rs.getDate("fechainiabstract"));
                congresomodelo.setFinalizacionabs(rs.getDate("fechafinabstract"));
                congresomodelo.setIdlistacorreo(rs.getInt("idlistacorreofk"));

                }
            } catch (SQLException e) {
                e.printStackTrace();
            }

                return congresomodelo;
        }


    public CCongresoModelo consultacongresomod(String acronimo)
        {
            CCongresoModelo congresomodelo = new CCongresoModelo();

            sqlstr = "SELECT * FROM congreso WHERE acronimo = '" + acronimo +
"'";
            bd.setConsulta(sqlstr);
            ResultSet rs = bd.getResultado();
            try
            {
                while (rs.next())
                {
```

```
                congresomodelo.setAcronimo(rs.getString("acronimo"));
                congresomodelo.setTema(rs.getString("tema"));
                congresomodelo.setEdicion(rs.getInt("edicion"));
                congresomodelo.setInicio(rs.getDate("fechaini"));
                congresomodelo.setFinalizacion(rs.getDate("fechafin"));
                congresomodelo.setDenominacion(rs.getString("denominacion"));
                congresomodelo.setPrecio(rs.getDouble("precio"));
                congresomodelo.setDireccion(rs.getString("direccion"));
                congresomodelo.setPais(rs.getString("pais"));
                congresomodelo.setUrl(rs.getString("web"));
                congresomodelo.setDescripcion(rs.getString("descripcion"));
                congresomodelo.setOrganizador(rs.getString("organizador"));
                congresomodelo.setViaje(rs.getString("viaje"));
                congresomodelo.setLenguaje(rs.getString("lenguaje"));
                congresomodelo.setCiudad(rs.getString("ciudad"));
                congresomodelo.setTelefono(rs.getString("telefono"));
                congresomodelo.setFaximil(rs.getString("fax"));
                congresomodelo.setEmail(rs.getString("email"));
                congresomodelo.setCodigo(rs.getString("cp"));
                congresomodelo.setInicioins(rs.getDate("fechainiins"));
                congresomodelo.setFinalizacionins(rs.getDate("fechafinins"));
                congresomodelo.setInicioabs(rs.getDate("fechainiabstract"));
                congresomodelo.setFinalizacionabs(rs.getDate("fechafinabstract"));
                congresomodelo.setIdlistacorreo(rs.getInt("idlistacorreofk"));

                }
            } catch (SQLException e) {
                e.printStackTrace();
            }

                return congresomodelo;
        }


public Collection consultacongresos()
        {
            Collection congresos = new Vector();
            sqlstr = "SELECT * FROM congreso";
            bd.setConsulta(sqlstr);
            ResultSet rs = bd.getResultado();
            try
            {
                while (rs.next())
                {
                CCongresoModelo congresomodelo = new CCongresoModelo();
                congresomodelo.setAcronimo(rs.getString("acronimo"));
                congresomodelo.setTema(rs.getString("tema"));
```

```
                congresomodelo.setEdicion(rs.getInt("edicion"));
                congresomodelo.setInicio(rs.getDate("fechaini"));
                congresomodelo.setFinalizacion(rs.getDate("fechafin"));
                congresomodelo.setDenominacion(rs.getString("denominacion"));
                congresomodelo.setPrecio(rs.getDouble("precio"));
                congresomodelo.setDireccion(rs.getString("direccion"));
                congresomodelo.setPais(rs.getString("pais"));
                congresomodelo.setUrl(rs.getString("web"));
                congresomodelo.setDescripcion(rs.getString("descripcion"));
                congresomodelo.setOrganizador(rs.getString("organizador"));
                congresomodelo.setViaje(rs.getString("viaje"));
                congresomodelo.setLenguaje(rs.getString("lenguaje"));
                congresomodelo.setCiudad(rs.getString("ciudad"));
                congresomodelo.setTelefono(rs.getString("telefono"));
                congresomodelo.setFaximil(rs.getString("fax"));
                congresomodelo.setEmail(rs.getString("email"));
                congresomodelo.setCodigo(rs.getString("cp"));
                congresomodelo.setInicioins(rs.getDate("fechainiins"));
                congresomodelo.setFinalizacionins(rs.getDate("fechafinins"));
                congresomodelo.setInicioabs(rs.getDate("fechainiabstract"));
                congresomodelo.setFinalizacionabs(rs.getDate("fechafinabstract"));
                congresomodelo.setIdlistacorreo(rs.getInt("idlistacorreofk"));
                congresos.add(congresomodelo);
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }

                return congresos;
        }
```

```
    public int altaCongreso(String acr,String tem,int edi,String den,String fini,String
ffin,Double pvp,String dir,String cp,String pais,String ciu,String url,String tel,String
fax,String email,String desc,String org,String viaje,String leng,String finiins,String
ffinins,String finiabs,String ffinabs)
    {
        sqlstr = "INSERT INTO
congreso(acronimo,tema,edicion,denominacion,fechaini,fechafin,precio,direccion,cp,p
ais,ciudad,web,telefono,fax,email,descripcion,organizador,viaje,lenguaje,fechainiins,fe
chafinins,fechainiabstract,fechafinabstract) VALUES('" + acr + "','" + tem + "'," + edi
+ ",'" + den + "','" + fini + "','" + ffin + "'," + pvp + ",'" + dir + "','" + cp + "','" +
pais + "','" + ciu + "','" + url + "','" + tel + "','" + fax + "','" + email + "','" + desc +
"','" + org + "','" + viaje + "','" + leng + "','" + finiins + "','" + ffinins + "','" + finiabs
+ "','" + ffinabs + "')";
        bd.setConsulta(sqlstr);
```

```
            y = bd.executeSQL();
            return y;
        }
        public int bajaCongreso(String[] acr)
        {
            for (int x=0;x<acr.length;x++)
            {
                sqlstr = "DELETE FROM congreso WHERE acronimo = '" + acr[x] + "'";
                bd.setConsulta(sqlstr);
                y = bd.executeSQL();
            }
            return y;
        }
        public int modificacionCongreso(String acr,String tem,int edi,String den,String
fini,String ffin,Double pvp,String dir,String cp,String pais,String ciu,String url,String
tel,String fax,String email,String desc,String org,String viaje,String leng,String
finiins,String ffinins,String finiabs,String ffinabs)
        {
            sqlstr = "UPDATE congreso SET tema = '" + tem + "',edicion = " + edi +
",denominacion = '" + den + "',fechaini = '" + fini + "',fechafin = '" + ffin + "',precio
= " + pvp + ",direccion = '" + dir + "',cp = '" + cp + "',pais = '" + pais + "',ciudad =
'" + ciu + "',web = '" + url + "',telefono = '" + tel + "',fax = '" + fax + "',email = '" +
email + "',descripcion = '" + desc + "',organizador = '" + org + "',viaje = '" + viaje +
"' ,lenguaje = '" + leng + "',fechainiins = '" + finiins + "',fechafinins = '" + ffinins +
"',fechainiabstract = '" + finiabs + "',fechafinabstract = '" + ffinabs + "' WHERE
acronimo = '" + acr + "'";
            bd.setConsulta(sqlstr);
            y = bd.executeSQL();
            return y;
        }
}
```

CBBDDBean.java

```
package javabean;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
/**
 * Creado por Marcos Patricio Amate
 * User: Marcos
 * Date: 26-abr-2005
 * Time: 20:13:57
 */
public class CBBDDBean
```

```
{
   private Connection conexion = null;
   private ResultSet resultado = null;
   private java.sql.Statement sentencia = null;
   private String datosConexion = "";
   private String servidorBDatos = "";
   private String nombreBDatos = "";
   private String puertoBDatos = "";
   private String usuarioBDatos = "";
   private String claveBDatos = "";
   private String consultaBDatos = "";
   private String valorCampo = "";
   private int x;

   public int executeSQL()
      {
         try
         {
            datosConexion = "jdbc:mysql://" + servidorBDatos + "/" +
nombreBDatos;
            Class.forName("org.gjt.mm.mysql.Driver").newInstance();
            conexion = DriverManager.getConnection(datosConexion.trim());
            sentencia = conexion.createStatement();
            x = sentencia.executeUpdate(consultaBDatos.trim());
         }catch(Exception e)
         {
            System.out.println("Error: " + e.getMessage());
         }finally
         {
            cerrarConexion();
            return x;
         }
      }

   public ResultSet getResultado()
   {
      try
      {
         datosConexion = "jdbc:mysql://" + servidorBDatos + "/" + nombreBDatos;
         Class.forName("org.gjt.mm.mysql.Driver").newInstance();
         conexion = DriverManager.getConnection(datosConexion.trim());
         sentencia = conexion.createStatement();
         resultado = sentencia.executeQuery(consultaBDatos.trim());
      }catch(Exception e)
      {
         System.out.println("Error: " + e.getMessage());
      }finally
```

```
        {
            cerrarConexion();
            return resultado;
        }
    }
    public void setServidor(String servidorBDatos)
    {
        this.servidorBDatos = servidorBDatos;
    }
    public void setNombre(String nombreBDatos)
    {
        this.nombreBDatos = nombreBDatos;
    }
    public void setPuerto(String puertoBDatos)
    {
        this.puertoBDatos = puertoBDatos;
    }
    public void setUsuario(String usuarioBDatos)
    {
        this.usuarioBDatos = usuarioBDatos;
    }
    public void setClave(String claveBDatos)
    {
        this.claveBDatos = claveBDatos;
    }
    public void setConsulta(String consultaBDatos)
    {
        this.consultaBDatos = consultaBDatos;
    }
    public void cerrarConexion()
    {
        try
        {
            conexion.close();
        }catch(Exception e)
        {
            System.out.println("Error: " + e.getMessage());
        }
    }
    public void setCampoBaseDatos(String campo)
    {
        try
        {
            valorCampo = resultado.getString(campo);
        }catch(Exception e)
        {
            valorCampo = "- no existe ese campo -";
```

```
            System.out.println("Error: " + e.getMessage());
        }
    }
    public String getValor()
    {
        return valorCampo;
    }

}
```

MODELO DE DATOS

```
package modelos;

import java.sql.Date;

/**
 * Creado por Marcos Patricio Amate
 * User: Marcos
 * Date: 28-abr-2005
 * Time: 8:43:19
 */
public class CCongresoModelo
{
    private String acronimo;
    private String tema;
    private int edicion;
    private Date inicio;
    private Date finalizacion;
    private String denominacion;
    private double precio;
    private String direccion;
    private String pais;
    private String url;
    private String descripcion;
    private String organizador;
    private String viaje;
    private String lenguaje;
    private String ciudad;
    private String telefono;
    private String faximil;
    private String email;
    private String codigo;
    private Date inicioins;
    private Date finalizacionins;
    private Date inicioabs;
```

```
    private Date finalizacionabs;
    private int idlistacorreo;
    private String dni;

    public String getDni() {
        return dni;
    }

    public void setDni(String dni) {
        this.dni = dni;
    }
    public String getAcronimo() {
        return acronimo;
    }

    public void setAcronimo(String acronimo) {
        this.acronimo = acronimo;
    }

    public String getTema() {
        return tema;
    }

    public void setTema(String tema) {
        this.tema = tema;
    }

    public int getEdicion() {
        return edicion;
    }

    public void setEdicion(int edicion) {
        this.edicion = edicion;
    }

    public Date getInicio() {
        return inicio;
    }

    public void setInicio(Date inicio) {
        this.inicio = inicio;
    }

    public Date getFinalizacion() {
        return finalizacion;
    }
```

```
    public void setFinalizacion(Date finalizacion) {
        this.finalizacion = finalizacion;
    }

    public String getDenominacion() {
        return denominacion;
    }

    public void setDenominacion(String denominacion) {
        this.denominacion = denominacion;
    }

    public double getPrecio() {
        return precio;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public String getPais() {
        return pais;
    }

    public void setPais(String pais) {
        this.pais = pais;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getDescripcion() {
        return descripcion;
    }
```

```
public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}

public String getOrganizador() {
    return organizador;
}

public void setOrganizador(String organizador) {
    this.organizador = organizador;
}

public String getViaje() {
    return viaje;
}

public void setViaje(String viaje) {
    this.viaje = viaje;
}

public String getLenguaje() {
    return lenguaje;
}

public void setLenguaje(String lenguaje) {
    this.lenguaje = lenguaje;
}

public String getCiudad() {
    return ciudad;
}

public void setCiudad(String ciudad) {
    this.ciudad = ciudad;
}

public String getTelefono() {
    return telefono;
}

public void setTelefono(String telefono) {
    this.telefono = telefono;
}

public String getFaximil() {
    return faximil;
```

```
    }

    public void setFaximil(String faximil) {
        this.faximil = faximil;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }

    public Date getInicioins() {
        return inicioins;
    }

    public void setInicioins(Date inicioins) {
        this.inicioins = inicioins;
    }

    public Date getFinalizacionins() {
        return finalizacionins;
    }

    public void setFinalizacionins(Date finalizacionins) {
        this.finalizacionins = finalizacionins;
    }

    public Date getInicioabs() {
        return inicioabs;
    }

    public void setInicioabs(Date inicioabs) {
        this.inicioabs = inicioabs;
    }

    public Date getFinalizacionabs() {
```

```
        return finalizacionabs;
    }

    public void setFinalizacionabs(Date finalizacionabs) {
        this.finalizacionabs = finalizacionabs;
    }
    public int getIdlistacorreo() {
        return idlistacorreo;
    }

    public void setIdlistacorreo(int idlistacorreo) {
        this.idlistacorreo = idlistacorreo;
    }

}
```

CAPA JSP

congreso.jsp

```
<%@ page import="modelos.CCongresoModelo,
                 java.util.*,
                 java.sql.Date"%>
<html>
    <head>
        <title>Seminaris i congressos</title>
        <meta http-equiv="content-type" content="text/html;charset=utf-8"></meta>
        <link rel="stylesheet" href="../css/agenda.css" type="text/css"></link>

    <body>
        <div id="pagewidth">
            <%@ include file="modulos/cabecera.jsp"%>
            <div id="outer">
                <div id="inner">
                    <div id="leftcol">
                        <%@ include file="modulos/usuario.jsp"%>
                        <%@ include file="modulos/menu.jsp"%>
                        <%@ include file="modulos/calendari.jsp"%>
                    </div>
                    <div id="maincol">
                        <h2>CONGRÉS:            <%Collection            congresos            =
(Collection)request.getAttribute("congreso");
                                    CCongresoModelo                congreso                =
(CCongresoModelo)congresos.iterator().next();
                                %><%=congreso.getDenominacion()%></h2>
                        <br>
                        <table width="100%" border="0">
                            <tr>
```

```
                        <td width="20%">Data d'inici</td>
                        <td width="20%">Data final</td>
                        <td width="60%">Títol</td>
                    </tr>
                    <tr>
                        <td width="20%"><%=congreso.getInicio()%></td>
                        <td width="20%"><%=congreso.getFinalizacion()%></td>
                        <td                                          width="60%"><a
href=""><%=congreso.getDenominacion()%></a></td>
                    </tr>
                </table>
                <table>
                    <tr>
                        <td></td>
                    </tr>
                    <tr>
                        <td></td>
                    </tr>
                    <tr>
                        <td width="100%">Adreça</td>
                    </tr>
                    <tr>
                        <td><%=congreso.getDireccion()%></td>
                    </tr>
                    <tr>
                        <td></td>
                    </tr>
                    <tr>
                        <td></td>
                    </tr>
                </table>
                <table>
                    <tr>
                        <td width="10%">Codi Postal</td>
                        <td width="15%">Telèfon</td>
                        <td width="15%">Fax</td>
                        <td width="15%">E-Mail</td>
                        <td width="30%">Ciutat</td>
                        <td width="15%">Pais</td>
                    </tr>
                    <tr>
                        <td width="10%"><%=congreso.getCodigo()%></td>
                        <td width="15%"><%=congreso.getTelefono()%></td>
                        <td width="15%"><%=congreso.getFaximil()%></td>
                        <td width="15%"><%=congreso.getEmail()%></td>
                        <td width="30%"><%=congreso.getCiudad()%></td>
                        <td width="15%"><%=congreso.getPais()%></td>
```

```
                    </tr>
                </table>
                <br>
                <br>
                <br>
                <br>
                <br>
                <br>

                <h2>DEADLINES</h2>

                <!--Coger fecha min y max de los deadlines y luego hacer el for para cada una de las fechas,
                llamando al include .jsp con los parámetros del mes y año que pertoquen-->
                <%

                // Calendar[] calendario = new GregorianCalendar[12];
                Date min = null;
                Date max = null;
                min = congreso.getInicio();
                if (min.after(congreso.getInicioins()) || min.after(congreso.getInicioabs()))
                {
                    if(min.after(congreso.getInicioins()))
                    {
                        min = congreso.getInicioins();
                    }
                    if(min.after(congreso.getInicioabs()))
                    {
                        min = congreso.getInicioabs();
                    }
                }
                max = congreso.getFinalizacion();
                if (max.before(congreso.getFinalizacionins()) || max.before(congreso.getFinalizacionabs()))
                {
                    if(max.before(congreso.getFinalizacionins()))
                    {
                        max = congreso.getFinalizacionins();
                    }
                    if (max.before(congreso.getFinalizacionabs()))
                    {
                        max = congreso.getFinalizacionabs();
                    }
                }
```

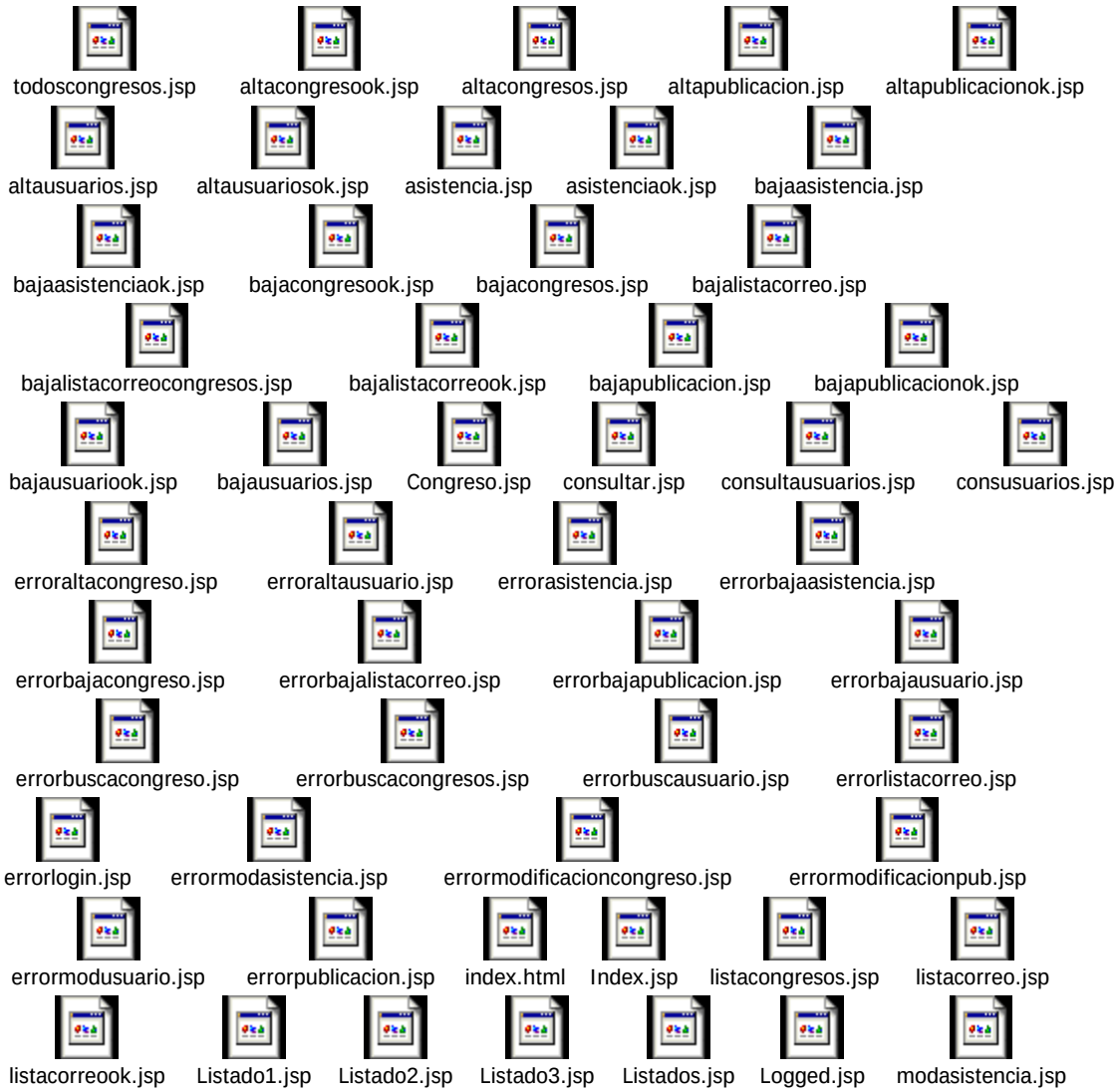
```
                    GregorianCalendar gMax = new GregorianCalendar();
                    gMax.setTime(max);
                    GregorianCalendar gMin = new GregorianCalendar();
                    gMin.setTime(min);
                    int t = gMax.get(Calendar.MONTH) - gMin.get(Calendar.MONTH);
                    //System.out.println("Gmin:" + gMin + " Gmax" + gMax);
                   // System.out.println(t);

                  for(int z=0;z<=t;z++)
                  {
                     %>
                     <jsp:include page="modulos/calendariocongreso.jsp"
                     flush="true">
                        <jsp:param                                    name="month"
value="<%=gMin.get(Calendar.MONTH)%>"/>
                        <jsp:param                                     name="year"
value="<%=gMin.get(Calendar.YEAR)%>"/>
                        </jsp:include><%
                     gMin.add(Calendar.MONTH,1); //mes++
                  }%>
                  <div id="footmain">
                  <table>
                     <tr>
                        <td><img
src="../imagenes/iniinscripcion.GIF"></img></td><td>Data d'inici Inscripció</td>
                        <td><img
src="../imagenes/fininscripcion.GIF"></img></td><td>Data                finalització
Inscripció</td>
                     </tr>
                     <tr>
                        <td><img
src="../imagenes/inipublicacion.GIF"></td></img><td>Data                    d'inici
Publicacions</td>
                        <td><img
src="../imagenes/finpublicacion.GIF"></td><td>Data finalització Publicacions</td>
                     </tr>
                     <tr>
                        <td><img
src="../imagenes/inicongreso.GIF"></td><td>Data d'inici Congrès</td>
                        <td><img
src="../imagenes/fincongreso.GIF"></td><td>Data finalització Congrès</td>
                     </tr>
                  </table>
                  <div class="clr"></div>
                  </div>
                  <div class="clr"></div>
               </div>
```

```
            </div>
            <div class="clr"></div>
          </div>
          <%@ include file="modulos/footer.jsp"%>
          <div class="clr">
          </div>
        </div>
    </body>
</html>
```

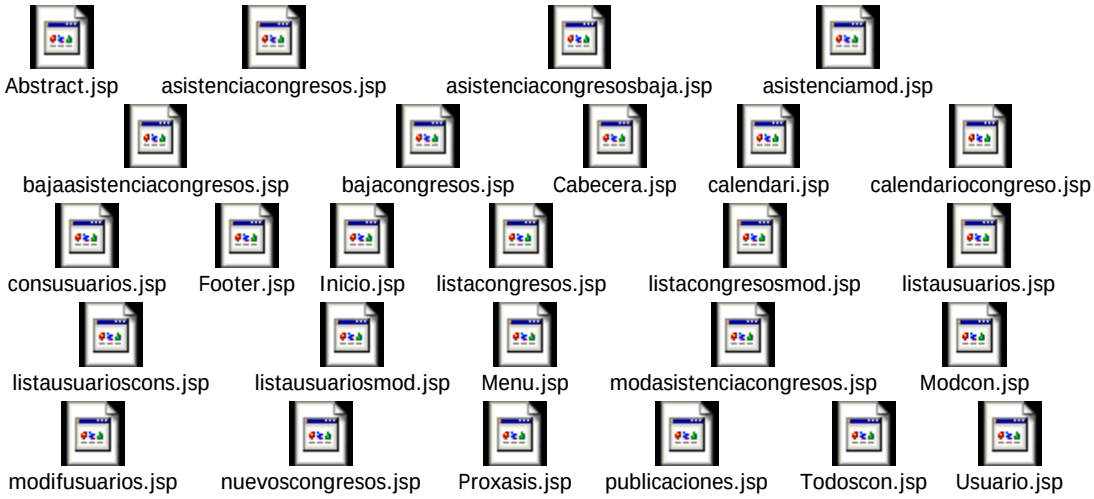
Con este código se ve el recorrido de todo el caso de uso de consulta de un cogreso. Mediante la separación de tres capas que arriba he mencionado.

5.3. Listado del código implementado

5.3.1.- Capa JSP



Módulos en Capa JSP



5.3.2.- Capa SERVLET

alta2congreso.java altaasistencia.java altacongreso.java altapublicacion.java altapublicacion2.java

altausuario.java anadirselista.java asistencia.java bajaanadirselista.java bajaasistencia.java

bajacongresos.java bajacongresos2.java bajalista.java bajapub.java bajapublicacion.java

bajausuarios.java consultacongreso.java consultacongresos.java consultausuarios.java

consultausuarioscons.java consultausuariosmod.java consusuario.java index.java

listabajaasistencia.java listamodasistencia.java login.java modasistencia.java modcongresos.java

modifasistencia.java modifcongreso.java modifusuario.java modpub.java modpub2.java

modpublicacion.java modusuarios.java proxasistencias.java publicacion.java todoscongresos.java

5.3.3.- Capa JAVA BEAN

CAsistenciaBean.java CBBDDBean.java CCongresoBean.java CListaBean.java CPublicacionBean.java

CUsuarioBean.java

Modelos de Datos

CAsistenciaModelo.java CCongresoModelo.java CListaModelo.java CPublicacionModelo.java

CUsuarioModelo.java

6.- Conclusiones

6.1.- Conclusión

La conclusión general que se ha obtenido es la realización de los objetivos propuestos

- Tenemos una aplicación útil que simplifica la tarea de gestionar un calendario de congresos además de ser totalmente independiente de la máquina y del sistema.
- He ampliado los conocimientos adquiridos durante la carrera, y han sido ampliados con contenidos muy actuales como la tecnología J2EE, para la realización de aplicaciones web.

Durante la carrera he estudiado las asignaturas que me permitían generar un proyecto íntegro de ingeniería de programación. Este Trabajo de Fin de Carrera, me ha aportado la posibilidad de verificar mis conocimientos en la generación de una aplicación desde cero hasta su totalidad, y además utilizando una tecnología muy potente y actual como Java.

El principal inconveniente que me he encontrado es la individualización del trabajo, y el tiempo. Es decir solo he sido yo el generador de toda la aplicación, cuando

normalmente un proyecto de similar envergadura podría ser generado por dos o tres personas, un equipo de trabajo.

Me he dado cuenta que he cometido un fallo en la planificación porque podría haber dedicado mes y medio a la implementación, ya que me han faltado un par de cosas que ahora comentaré en las mejoras del proyecto.

6.2.- Mejoras

- El calendario que sale en el menú de la aplicación funciona y de hecho se puede actualizar e ir a cualquier mes de cualquier año. Pero la funcionalidad que iba a darle en un principio era que iban a aparecer en las fechas (mediante colores por temas) los inicios de los congresos, esta parte final no funciona, ahora es estática (siempre se dibujan los colores en las mismas posiciones), no funciona realmente por falta de tiempo, ya que es donde me he quedado actualmente trabajando.

- En el apartado de `consultar congreso', la búsqueda de Congresos por palabras clave tampoco funciona, de hecho iba a comenzarla al acabar el calendario del menú. Ahora mismo busca por denominación de congreso, por el título, y así sí que funciona.

- La asistencia a un congreso, sería mejorable. Lo que realizo es una búsqueda de todos los congresos que tienen una fecha posterior a la del sistema, y se devuelve una colección de congresos. Cuando se selecciona una asistencia sin ponencia a veces falla. Lo he estado revisando pero tampoco me ha dado tiempo.

- En los campos de fecha sería conveniente bloquearlos y que apareciera un botón al lado que al pulsarlo se activaría una ventana emergente con un calendario, para que el usuario pudiera escoger la fecha y de esta manera la fecha pasaría al campo mencionado. Así se evitarían posibles errores de validación de fechas.

- Opción de cerrar sesión, añadiría un cierre de sesión para que se pueda cambiar de usuario, sin la necesidad de cerrar el navegador.

Estas son las mejoras que yo propondría, aunque de hecho se pueden mejorar muchas más. Pero si estas funcionalidades se arreglan la aplicación funcionaría a pleno rendimiento al 100%. Si no las he podido realizar bien han sido por falta de tiempo, y como predije en su momento me ha llevado más tiempo del necesario realizar la implementación, debido a la falta de experiencia para la realización de este tipo de aplicaciones.