Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Ingeniería en Computación

PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLÁSICAS

Reporte Final de PT02

Alumno:
Roberto Clemente González Rubio
204301921
Asesorado por:
M.en C. Rafaela Blanca Silva López
No. Económico: 17114

Trimestre 12-P
28 de Agosto de 2012

Índice

1 Objetivos del Sistema

1.1 Objetivo General

Construir un Sistema de Información el cual facilite el aprendizaje a médicos residentes en el tratamiento de enfermedades Neoplásicas.

1.2 Objetivos Específicos

a. Analizar las Necesidades de Aprendizaje de los Médicos Residentes en el Seguimiento del Tratamiento de Enfermedades Neoplásicas.
b. Investigar acerca de herramientas y tecnologías para sistemas de aprendizaje en línea.
c. Diseñar el Sistema de Aprendizaje para Médicos Residentes en el Seguimiento del Tratamiento de Enfermedades Neoplásicas que cubra los siguientes módulos:
    - Consulta de contenidos de enfermedades Neoplásicas
    - Seguimiento de casos de pacientes con enfermedades Neoplásicas
    - Evaluación del aprendizaje
g. Diseñar el flujo de trabajo electrónico para el seguimiento de pacientes con enfermedades Neoplásicas.
h. Implementar el Sistema de Aprendizaje para Médicos Residentes en el Seguimiento del Tratamiento de Enfermedades Neoplásicas.
i. Realizar pruebas para la liberación del Sistema de Aprendizaje para Médicos Residentes en el Seguimiento del Tratamiento de Enfermedades Neoplásicas.
j. Elaborar la documentación del Sistema de Aprendizaje para Médicos Residentes en el Seguimiento del Tratamiento de Enfermedades Neoplásicas.
k. Integrar el Sistema de Aprendizaje para Médicos Residente en el Seguimiento del Tratamiento de Enfermedades Neoplásicas, en los Ambientes Virtuales de Aprendizaje E-Learning –Knowledge.

2 Antecedentes

En la actualidad existen diversos Sistemas de Información o herramientas los cuales pretenden facilitar el aprendizaje en línea a médicos residentes.

Uno es la herramienta “Autor *Vértice 2.1*” el cual es un software con el que se pueden diseñar contenidos multimedia e interactivos, sin necesidad de conocimientos de programación o diseño web, (referencia) y aunque su manejo es sencillo e intuitivo, pareciera más un editor HTML. Para realizar una herramienta de evaluación se tiene que empezar desde cero. Aunque tiene la ventaja de poder insertarse fácilmente en Moodle o a algún otro sistema similar y la desventaja de que requiere de algún sistema (host) para poder observar su funcionamiento, ya que su contenido es puramente HTML.

Otro es el Sistema llamado CASUS. Este Sistema tiene un diseño fácil de usar y no requiere que los autores de los casos tengan un elevado conocimiento informático.

El Sistema CASUS permite la creación, evaluación y resolución de casos vía Internet, posibilitando la incorporación de fotografías, cuadros, gráficos y películas de manera sencilla.

Los resultados obtenidos por los estudiantes son evaluados, midiendo el tiempo que han invertido en la realización de cada caso, el porcentaje de preguntas contestadas correctamente, etc.

En cada uno de los casos, los estudiantes van a disponer de la exploración del paciente y de la historia clínica, incluyendo la historia clínico-laboral. Esto permitirá al estudiante llegar a un diagnóstico de la patología del paciente. Además cada caso incorpora la posibilidad de usar unas opciones adicionales como, la opinión del experto “expert button” o hipervínculos “hiperlink”. Con ellas, los estudiantes van a obtener una información extra y complementaria de cada una de las técnicas diagnosticas o información adicional sobre cierta patología.

Como la interacción es un factor importante para la buena aceptación de esta herramienta de aprendizaje, los estudiantes son evaluados con una serie de cuestiones de diferentes formato (como elección múltiple, respuestas cortas…etc.) a lo largo del caso. Al final de cada una de estas cuestiones el estudiante puede comprobar el resultado de sus respuestas y disponer de la contestación correcta ampliamente comentada, para así poder conocer si su enfoque del caso está siendo correcto.

Otro Sistema cuyo objetivo es facilitar el aprendizaje en línea es el Sistema llamado LAMS (Learning Management Systems). Básicamente es un sistema o herramienta informática que permite a los agentes involucrados en la enseñanza, definir procesos para complementar el aprendizaje de sus alumnos. No sólo se trata de hacer ejercicios. Con LAMS se da un paso más: se crean y gestionan secuencias de actividades.

LAMS es una herramienta que permite desarrollar su trabajo a los distintos agentes involucrados en el proceso enseñanza – aprendizaje. Desde el punto de vista de LAMS, estos agentes o roles son cuatro:

- Autor
- Estudiante
- Administrador
- Monitor

Para cada uno de estos roles, LAMS proporciona unas herramientas de trabajo. En realidad es la misma pero el usuario sólo verá las opciones que necesita para llevar a cabo sus tareas.

El Autor dispone de una serie de actividades para componer las secuencias. Las actividades disponibles son: Anotación, cartelera, chat, compartir recursos, encuestas, enviar archivos, foro de discusión, opción múltiple, preguntas y respuestas, recursos, votación.

De igual forma, los estudiantes tendrán otra herramienta disponible en la que verán las actividades que tienen que realizar, el avance que van haciendo, las notas que los profesores le hacen y acciones

similares. El administrador y el monitor también tendrán sus propias herramientas de trabajo.

Tiene la ventaja de tener módulos específicos para hacer la integración con otros LMS , como Moodle, Sakai, Blackboard y dotLRN. Aparte de ser software libre bajo licencia GPL (General Public License).

3 Justificación

¿Qué son las enfermedades neoplásicas?

Las enfermedades neoplásicas son aquellas que surgen como resultado de una alteración en las células de los tejidos. Esta alteración está relacionada con los sistemas que regulan la reproducción y la diferenciación celular. Cuando estos sistemas fallan en una célula o en un grupo de células, estas se reproducen sin control y pierden además su especificidad, dando como resultado la aparición de tumores.[5]

Actualmente, dentro del proceso de enseñanza-aprendizaje una de las etapas más importantes es la etapa de evaluación. No solo se evalúa al alumno para certificar el nivel de conocimientos adquiridos a lo largo de todo el proceso, sino que también se evalúa para ofrecerle una retroalimentación sobre su aprendizaje.

La formación online supera a otros tipos de formación tradicionales porque en ella se pueden evaluar, incluso en muchos casos de forma automática, aspectos como la asistencia, las aportaciones o participaciones, los conocimientos o el proceso formativo en su totalidad.

Este tipo de formación tiene posibilidades casi ilimitadas para realizar la evaluación. Tanto en las plataformas como en los contenidos online se pueden incluir herramientas de evaluación interactivas y dinámicas que ofrecen por un lado, un feedback inmediato al alumno sobre los resultados alcanzados, y por otro lado, permiten a los gestores de la formación disponer de datos cuantitativos generados automáticamente por el sistema, que facilitan enormemente la tarea de evaluar.

A pesar de estas facilidades tampoco hay que descartar en muchos casos la intervención y juicio de un docente que evalúe aspectos cualitativos, a través de la actuación del alumno en los distintos contextos.

Se pueden definir principalmente 2 tipos de técnicas de evaluación:

- Técnicas objetivas (valoración cuantitativa)
- Técnicas subjetivas (valoración cualitativa)

Dentro de las técnicas objetivas podemos encontrar cuestionarios, los cuales miden el nivel de conocimientos del alumno. Existen diferentes instrumentos que utilizan esta técnica para evaluar al alumno, como son: Elección múltiple, Doble Alternativa (verdadero o falso), Asociación Parejas, Rellenar Huecos, Ordenar, Identificar, Clasificar, etc. También podemos encontrar a los ejercicios interactivos, los cuales pueden evaluar habilidades cognitivas o conocimientos del alumno. Existen

diferentes instrumentos que utilizan está técnica para evaluar al alumno, como son: Los Crucigramas, las sopas de letras, los rompecabezas o puzzles, etc.

En las técnicas subjetivas se incluyen todas las pruebas o técnicas que implican el juicio de un tutor o docente y son especialmente útiles para valorar múltiples aspectos del alumno, como su forma de expresarse, iniciativa, capacidad de afrontar problemas, habilidades sociales, etc. Éstas técnicas son las mismas que en la formación tradicional, aunque si existen ciertas mejoras o novedades. Por ejemplo, se puede diseñar un caso práctico dinámico cuyas respuestas abiertas y elaboradas sean enviadas directamente al buzón de correo del tutor, o se puede dotar al alumno de una serie de herramientas para trabajar colaborativamente y que la actuación del grupo sea registrada por el sistema: aportaciones, mensajes enviados, archivador electrónico, etc. Existen diferentes instrumentos qué utilizan esta técnica para evaluar al alumno, tales como: Exposición oral y redacción escrita, resolución de problemas, dinámicas de grupo, proyecto de fin de curso y casos prácticos.

Un caso práctico se utiliza para analizar una situación o problemática ya dada o que puede darse, partiendo de los diferentes factores involucrados, el análisis de los antecedentes, condiciones de la situación, etc.

Esta técnica pretende que el alumno reflexione sobre las distintas situaciones que pueden plantearse en la vida real acerca de la materia formativa que está recibiendo. Por ello, su objetivo fundamental es plantear a los alumnos situaciones conflictivas, para que éstos desarrollen su capacidad reflexiva, así como que sean capaces de tomar decisiones acerca de la mejor solución del problema o los problemas planteados.

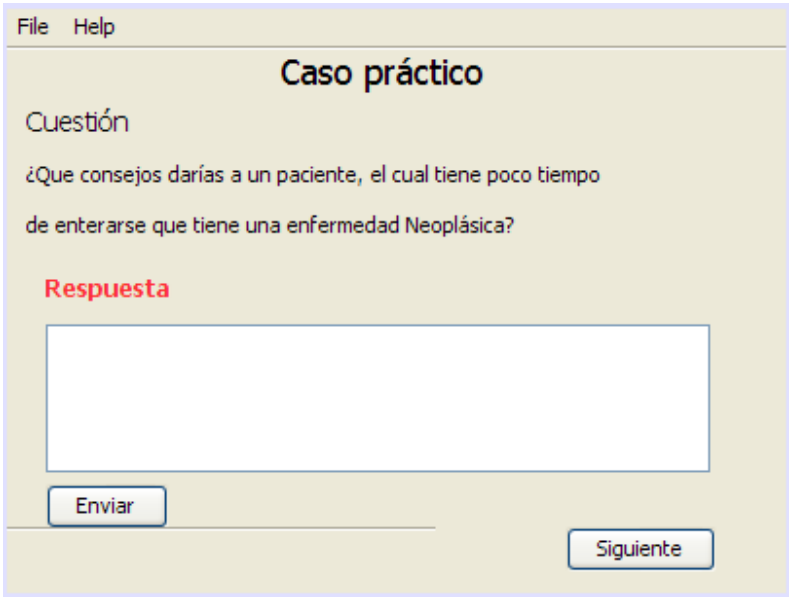


Figura 3.1 Ejemplo de Caso Práctico

Los alumnos deberán trabajar con él hasta llegar a asumirlo y comprenderlo en su totalidad. El conocimiento del ámbito (lectura previa de un tema) donde se inserta el caso, es supuesto básico para la eficacia de esta técnica.

El tutor deberá evaluar aspectos como son la participación de los alumnos, el nivel de profundización

de las ideas y la aplicabilidad de las soluciones ofrecidas, la creatividad, la capacidad de resolución de problemas, el número y frecuencia de las respuestas, etc.

Un caso práctico requiere de una preparación, individual o en grupo, y de una discusión colectiva, donde cada uno expone sus puntos de vistas y aporta sus soluciones a las cuestiones planteadas, con la guía del profesor o tutor.

El aprendizaje basado en la resolución de casos ha demostrado ser un método eficaz para el desarrollo de competencias en diversas ramas de la docencia en Medicina. Este método está todavía más indicado cuando existen dificultades para el desarrollo de un sistema de prácticas que permita a todos los estudiantes asistir a la clínica, o para ampliar competencias que han recibido escasa atención en el plan general de estudios, pero que van a ser necesarias en el quehacer diario de la práctica médica.

La elaboración de un Sistema de resolución de casos ameno, contribuye a la motivación del estudiante, quién se verá reforzado al comprobar sus avances o al corregir sus errores en el mismo momento en que realiza su práctica.

Por otra parte, el estudiante contribuye con sus opiniones a la mejora continua del sistema.

En la actualidad existen diversas dificultades que detienen el aprendizaje de los médicos residentes:

- La dificultad de poder realizar prácticas clínicas. Aunque se trata de uno de los sistemas preferidos por los residentes y más efectivos de aprendizaje, está a menudo muy limitado.
- El número de residentes muy elevado, en comparación con el número de médicos adscritos, lo que limita aún más estás prácticas.
- Solo unas pocas cuestiones van a ser evaluadas en los exámenes finales.

Debido a esto, el interés mostrado en algunos residentes suele ser bajo. Si bien se ha demostrado que la oportunidad de realizar un aprendizaje basado en la resolución de casos es acogido como un método muy eficaz, tanto para el aprendizaje en sí mismo, como para la motivación del estudiante.

Debido a esto se propone la creación de un Sistema de evaluación para médicos residentes en el seguimiento de enfermedades Neoplásicas bajo licencia Creative Commons (Algunos derechos reservados). El cual facilite a los Médicos (profesores) el proceso de evaluación de los residentes (alumnos), y a los residentes el proceso de aprendizaje.

La implementación de este Sistema de evaluación optimizara el proceso enseñanza – aprendizaje entre Médicos Adscritos y Médicos Residentes. Ahorrando tiempo a los residentes, ya que solo se requerirá básicamente de una computadora con un navegador WEB actual (ej. Mozilla Firefox 3.0) y acceso a Internet. Y ahorrando trabajo a los médicos adscritos, ya que desde cualquier computadora con un navegador WEB actual y acceso a Internet puede crear nuevos casos de pacientes, verificar las calificaciones de sus alumnos, etc.

El Sistema que se pretende realizar es parecido al Sistema CASUS antes mencionado, con la diferencia de que este Proyecto Terminal contará con un módulo de gestión de contenidos y se creará bajo licencia libre (Creative Commons).

3.1 Relación con Ingeniería en Computación

La ingeniería provee conocimientos específicos para la solución de problemas, por ello la carrera de Ingeniería en Computación impartida en la Universidad Autónoma Metropolitana – Azcapotzalco proporciona al alumno conocimientos avanzados en diversas UEA´s de Computación para la resolución de problemas, para la elaboración de este Proyecto Terminal se requieren conocimientos avanzados de Bases de Datos, Programación Orientada a Objetos e Ingeniería de Software para la documentación y realización correcta del Proyecto. Estos conocimientos pueden ser adquiridos en las UEA's: Bases de Datos, Estructura de Datos con Programación orientada a Objetos, Metodologías de Análisis Y Diseño de Sistemas de Información e Ingeniería de Software respectivamente.

4 Descripción Técnica

El proyecto considera los siguientes escenarios:

a) Módulo de gestión de contenidos
b) Módulo de gestión de casos
c) Módulo de evaluación

4.1 a) Módulo de Gestión de Contenidos

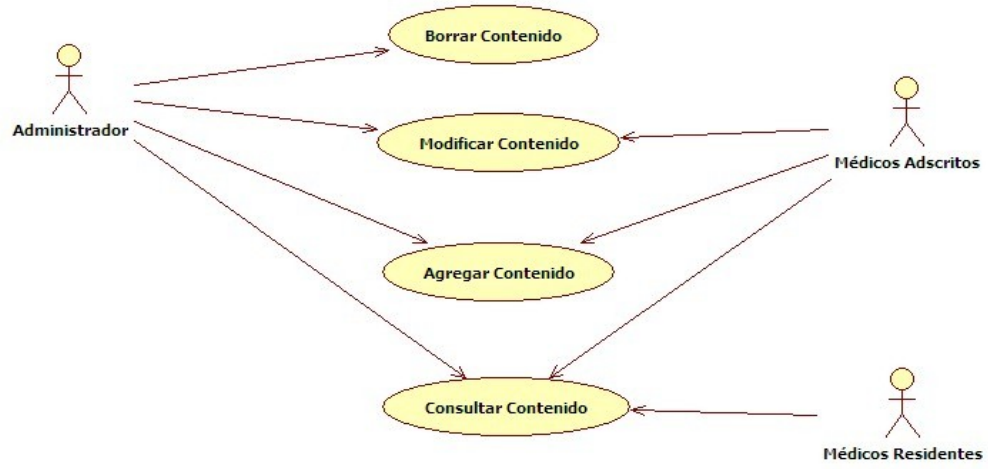


Figura 4.1.1 Caso de Uso módulo "a"

El objetivo del módulo de gestión de contenidos es almacenar la información sobre ciertas enfermedades Neoplásicas, por ejemplo, ¿Qué es cierta enfermedad?, ¿Cómo se adquiere?, Si es el caso ¿Cómo se cura?, ¿Qué tratamientos existen?, etc. Esto para que los Médicos Residentes antes o en la elaboración de un examen práctico, puedan consultar estos datos, debido a falta de experiencia de los Médicos Residentes, estos no podrán agregar información o modificarla, solo podrán consultarla. En

cambio los Médicos Adscritos, debido a su experiencia, podrán dar de alta información, modificarla y consultarla. El Administrador podrá dar de alta información, modificarla, consultarla o darla de baja. Cabe la posibilidad de que un Médico Adscrito sea del tipo Administrador.

4.2 b) Módulo de Gestión de Casos

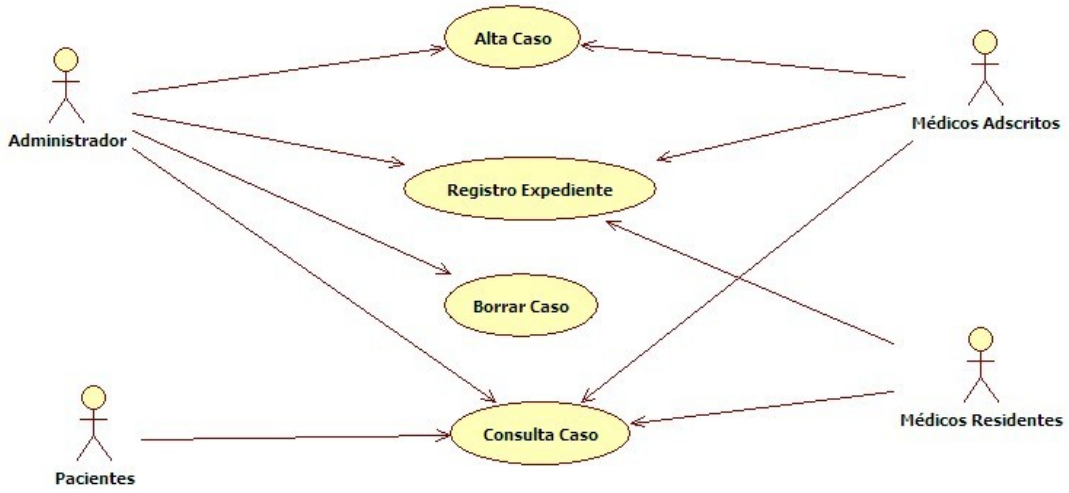


Figura 4.2.1 Caso de Uso módulo “b”

El objetivo de este módulo almacenar los casos reales y la historia clínica de pacientes con enfermedades Neoplásicas, elaborados previamente por un Médico Adscrito, por ejemplo, Dar de alta la historia clínica de un paciente, ¿Qué tratamiento se le asigno?, Si es el caso ¿Cómo fue curado?, en otro caso ¿Por qué no fue curado?, etc. Esto para que los médicos residentes al realizar una evaluación con casos prácticos puedan consultar esta información como apoyo.

En éste módulo los pacientes solo podrán consultar su caso (historia clínica, tratamiento, etc.), no podrán dar de alta casos o borrarlos. Los médicos residentes solo podrán consultar estos casos como apoyo académico. Los Médicos Adscritos debido a su experiencia podrán dar de alta casos y consultarlos. El Administrador podrá dar de alta casos, borrarlos y consultarlos. Cabe la posibilidad de que un Médico Adscrito sea del tipo Administrador.

4.3 c) Módulo de Evaluación

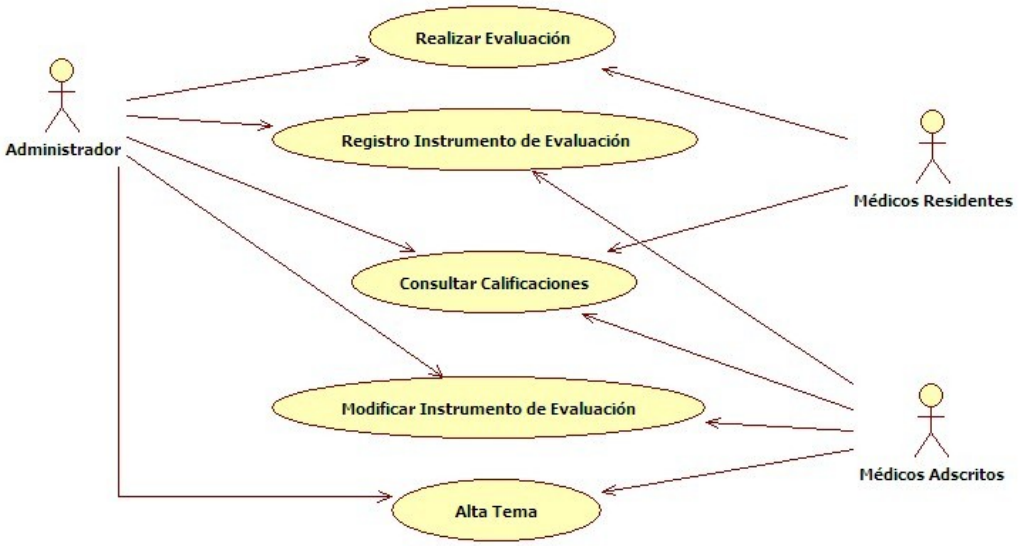


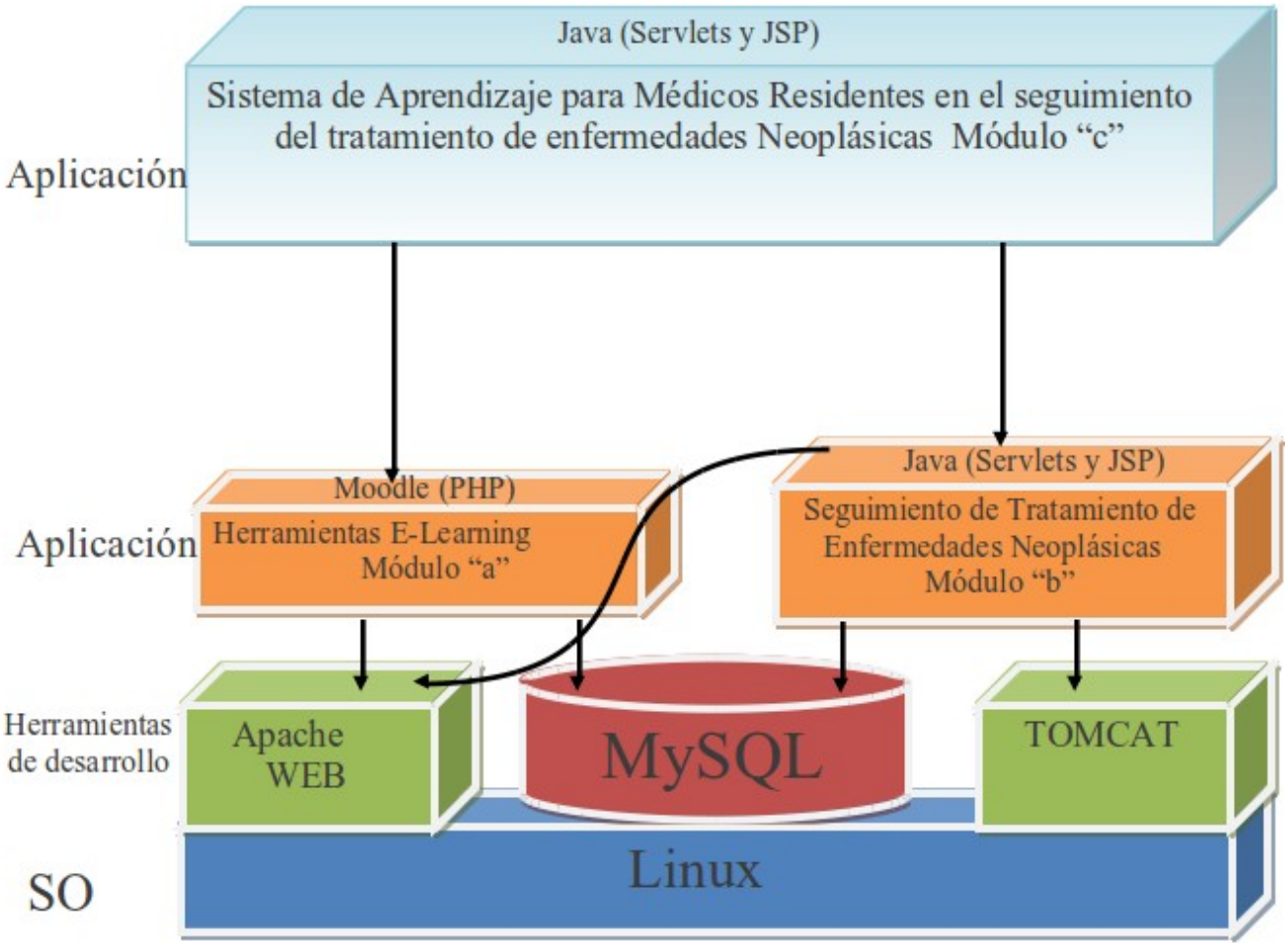
Figura 4.3.1 Caso de Uso módulo “c”

El objetivo principal de este módulo es, como su nombre lo indica, realizar evaluaciones mediante casos prácticos (reales) a médicos residentes. Los Médicos Residentes solo podrán realizar evaluaciones y consultar sus calificaciones. Los Médicos Adscritos podrán consultar las calificaciones de los Médicos Residentes, así como registrar Instrumentos de evaluación (exámenes con casos prácticos) o modificarlos y dar de alta temas nuevos. El Administrador podrá: consultar calificaciones, realizar evaluación, registrar y modificar instrumentos de evaluación y dar de alta temas nuevos.

5 Especificaciones Técnicas

- Plataforma Linux
- Tecnologías Java para el desarrollo de aplicaciones WEB
    - JSP
    - Servlets
- Herramientas para sistemas de aprendizaje en línea
    - Moodle
    - Manejador de Base de Datos MySQL
    - Servidor WEB Apache
    - Servidor de aplicaciones TOMCAT

A continuación se incluye el diagrama a bloques con las especificaciones técnicas:



6 Entregables

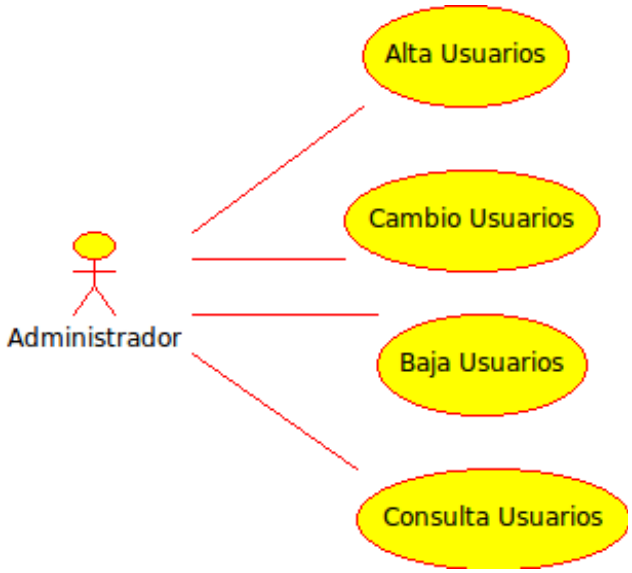
Los entregables para este proyecto son:

- Código fuente y compilado de la aplicación.
- Diagramas UML de casos de uso, clases y navegación; así como diagramas UML que documenten decisiones de diseño e implementación importantes.
- Diccionario de datos.
- Manual de usuario.
- Documentación de referencia.

7 Modelado del Sistema

7.1 Casos de Uso

7.1.1 Usuarios



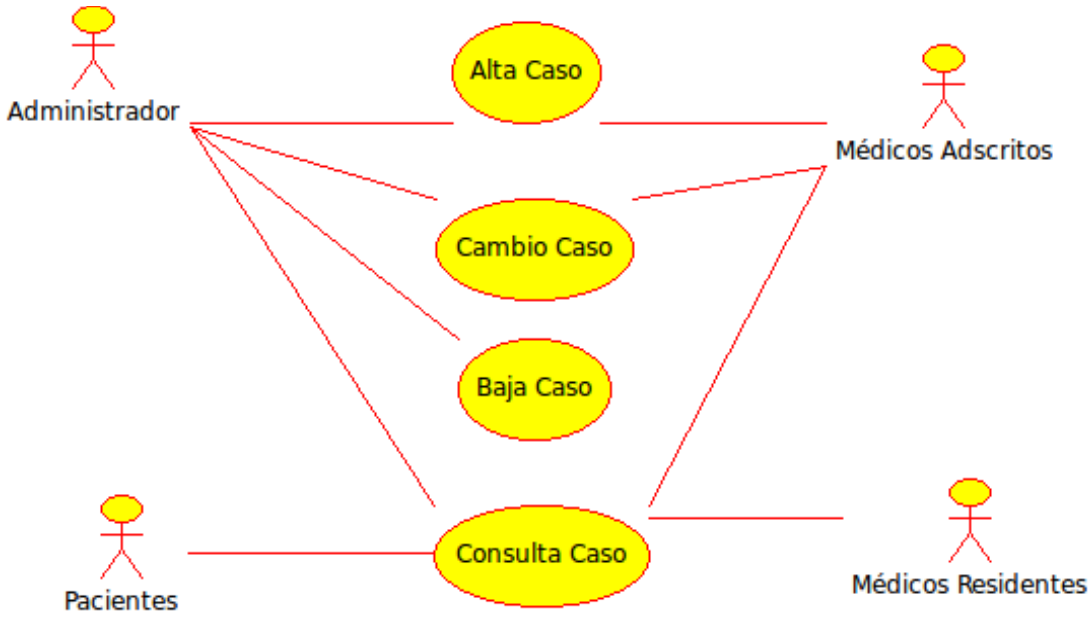
Autor:	Roberto González
Caso de Uso:	Alta Usuarios
Descripción:	Modelar los escenarios que contempla el registro de nuevos usuarios al sistema
Actores:	Administrador. Administrador que desee dar de alta un usuario.
Precondiciones:	Que se haya logueado correctamente el Administrador.
Escenario Principal:	El Administrador ingresa a la forma para dar de alta un usuario El Administrador llena los campos correspondientes del usuario
Escenario Alternativo:	ninguno
Poscondiciones:	Se da de alta al usuario en la base de datos

Autor:	Roberto González
Caso de Uso:	Cambio Usuarios
Descripción:	Modelar los escenarios que contempla la modificación de usuarios al sistema
Actores:	Administrador. Administrador que desee modificar a un usuario.
Precondiciones:	Que exista el usuario a modificar en el sistema.
Escenario Principal:	El Administrador ingresa a la forma para modificar al usuario El Administrador llena los campos correspondientes del usuario
Escenario Alternativo:	ninguno
Poscondiciones:	Se modifica al usuario en la base de datos

Autor:	Roberto González
Caso de Uso:	Baja Usuarios
Descripción:	Modelar los escenarios que contempla la borrado de usuarios al sistema
Actores:	Administrador. Administrador que desee borrar a un usuario.
Precondiciones:	Que exista el usuario a modificar en el sistema.
Escenario Principal:	El Administrador ve un listado de usuarios. El Administrador elige el usuario a borrar.
Escenario Alternativo:	ninguno
Poscondiciones:	Se borrar al usuario de la base de datos

Autor:	Roberto González
Caso de Uso:	Consulta Usuarios
Descripción:	Modelar los escenarios que contempla la consulta de usuarios al sistema
Actores:	Administrador. Administrador que desee consultar a un usuario.
Precondiciones:	Que exista el usuario a consultar en el sistema.
Escenario Principal:	El Administrador ve un listado de usuarios El Administrador ingresa a los datos detallados del usuario.
Escenario Alternativo:	ninguno
Poscondiciones:	El Administrador observa los datos detallados del usuario.

7.1.2 Casos



Autor:	Roberto González
Caso de Uso:	Alta Caso
Descripción:	Modelar los escenarios que contempla la inserción de nuevos casos al sistema.
Actores:	Administrador. Administrador que desee agregar un nuevo caso. Médico Adscrito. Médico Adscrito que desee agregar un nuevo caso.
Precondiciones:	Que se haya logueado correctamente el Administrador o el Médico Adscrito.
Escenario Principal:	El Administrador ingresa a la forma para modificar al usuario El Administrador llena los campos correspondientes del usuario
Escenario Alternativo:	ninguno
Poscondiciones:	Se da de alta el nuevo caso en la base de datos

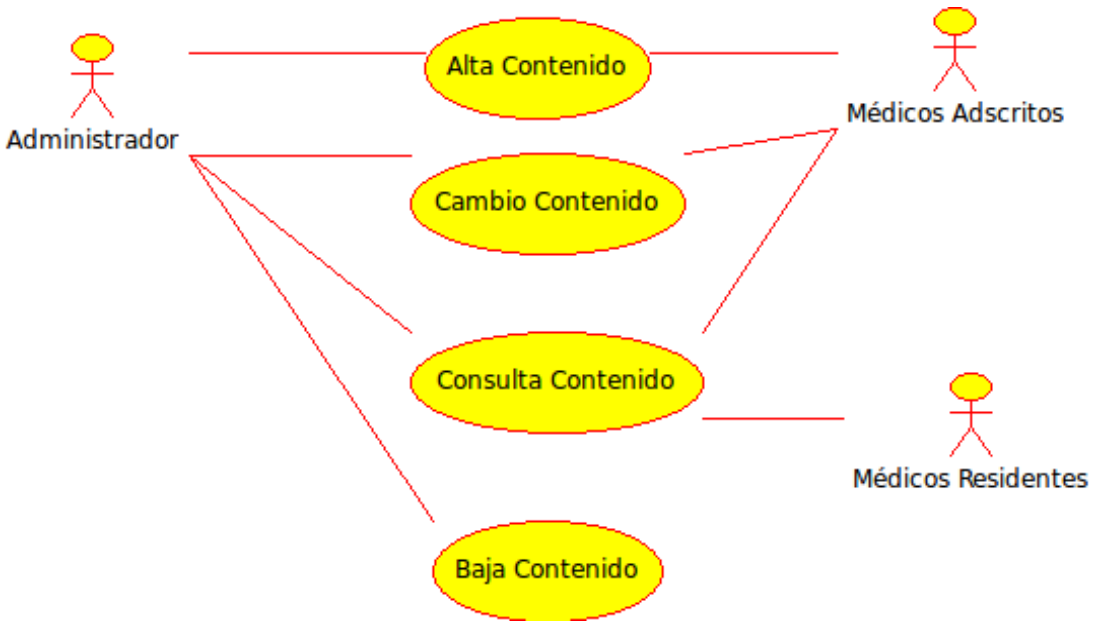
Autor:	Roberto González
Caso de Uso:	Cambio Caso
Descripción:	Modelar los escenarios que contempla la modificación de casos en el sistema.
Actores:	Administrador. Administrador que desee modificar un caso. Médico Adscrito. Médico Adscrito que desee modificar un caso.
Precondiciones:	Que exista el caso a modificar en la base de datos.
Escenario Principal:	El Administrador ingresa a la forma para modificar el caso. El Administrador llena los campos correspondientes del caso.
Escenario Alternativo:	ninguno
Poscondiciones:	Se modifica el caso en la base de datos.

Autor:	Roberto González
Caso de Uso:	Baja Caso
Descripción:	Modelar los escenarios que contempla el borrado de casos en el sistema.
Actores:	Administrador. Administrador que desee eliminar un caso.
Precondiciones:	Que exista el caso a borrar en la base de datos.
Escenario Principal:	El Administrador ve un listado de casos. El Administrador elige el caso a borrar de la base de datos.
Escenario Alternativo:	ninguno
Poscondiciones:	Se elimina el caso elegido de la base de datos.

Autor:	Roberto González
Caso de Uso:	Consulta Caso
Descripción:	Modelar los escenarios que contempla la consulta de casos en el sistema.
Actores:	Administrador. Administrador que desee consultar un caso. Médico Adscrito. Médico Adscrito que desee consultar un caso. Médico Residente: Médico Residente que desee consultar un caso. Paciente: Paciente que desee consultar un caso.
Precondiciones:	Que exista el caso a consultar en la base de datos.
Escenario Principal:	El Administrador, Médico Adscrito, Médicos Residente y Paciente ve un listado de casos. El Administrador, Médico Adscrito, Médicos Residente y Paciente eligen un

	caso a consultar a detalle.
Escenario Alternativo:	ninguno
Poscondiciones:	El Administrador, Médico Adscrito, Médicos Residente y Paciente ve a detalle el caso que consultó.

7.1.3 Contenidos



Autor:	Roberto González
Caso de Uso:	Alta Contenido
Descripción:	Modelar los escenarios que contempla la inserción de nuevos contenidos en el sistema.
Actores:	Administrador. Administrador que desee insertar un nuevo contenido en el sistema. Médico Adscrito. Médico Adscrito que desee insertar un nuevo contenido en el sistema.
Precondiciones:	Que se haya logueado correctamente el Administrador o el Médico Adscrito.
Escenario Principal:	El Administrador, Médico Adscrito, ingresa a la forma de inserción de nuevos contenidos.

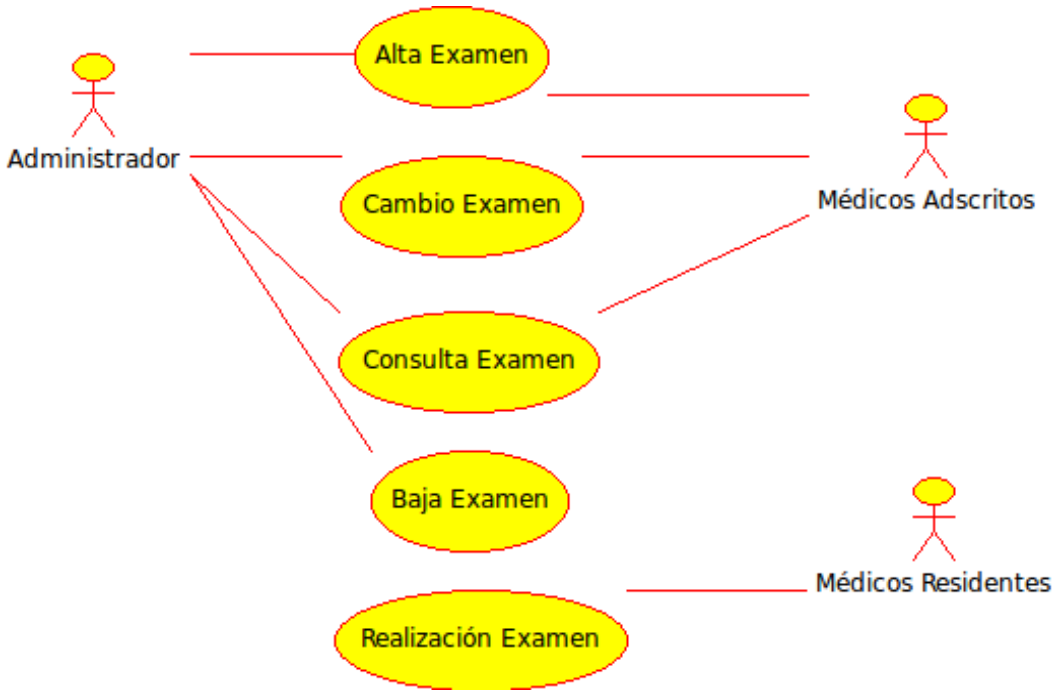
	El Administrador, Médico Adscrito, llena la forma de inserción de nuevos contenidos.
Escenario Alternativo:	ninguno
Poscondiciones:	El nuevo contenido se da de alta en la base de datos.

Autor:	Roberto González
Caso de Uso:	Cambio Contenido
Descripción:	Modelar los escenarios que contempla el cambio de contenidos en el sistema.
Actores:	Administrador. Administrador que desee modificar un contenido en el sistema. Médico Adscrito. Médico Adscrito que desee modificar un contenido en el sistema.
Precondiciones:	Que exista el contenido a modificar en la base de datos.
Escenario Principal:	El Administrador, Médico Adscrito, ingresa a la forma de cambio de contenidos. El Administrador, Médico Adscrito, llena la forma de cambio de contenidos.
Escenario Alternativo:	ninguno
Poscondiciones:	El contenido se modifica en la base de datos.

Autor:	Roberto González
Caso de Uso:	Consulta Contenido
Descripción:	Modelar los escenarios que contempla la consulta de contenidos en el sistema.
Actores:	Administrador. Administrador que desee consultar un contenido en el sistema. Médico Adscrito. Médico Adscrito que desee consultar un contenido en el sistema. Médico Residente. Médico Residente que desee consultar un contenido en el sistema.
Precondiciones:	Que exista el contenido a consultar en el sistema.
Escenario Principal:	El Administrador, Médico Adscrito y Médico Residente ve un listado de contenidos. El Administrador, Médico Adscrito y Médico Residente elige un contenido para verlo a detalle.
Escenario Alternativo:	ninguno
Poscondiciones:	El Administrador, Médico Adscrito y Médico Residente observa los detalles del contenido que consultó.

Autor:	Roberto González
Caso de Uso:	Baja Contenido
Descripción:	Modelar los escenarios que contempla la eliminación de contenidos en el sistema.
Actores:	Administrador. Administrador que desee eliminar un contenido en el sistema.
Precondiciones:	Que exista el contenido a eliminar en la base de datos.
Escenario Principal:	El Administrador, ve un listado de contenidos. El Administrador, elige un contenido a eliminar.
Escenario Alternativo:	ninguno
Poscondiciones:	El contenido se elimina de la base de datos.

7.1.4 Exámenes



Autor:	Roberto González
Caso de Uso:	Alta Examen
Descripción:	Modelar los escenarios que contempla la inserción de exámenes en el sistema.
Actores:	Administrador. Administrador que desee insertar un examen en el sistema. Médico Adscrito. Médico Adscrito que desee insertar un examen en el sistema.

Precondiciones:	Que se haya logueado correctamente el Administrador o el Médico Adscrito.
Escenario Principal:	El Administrador, Médico Adscrito, ingresa a la forma de alta de exámenes. El Administrador, Médico Adscrito, llena la forma de alta de exámenes.
Escenario Alternativo:	ninguno
Poscondiciones:	El examen se inserta en la base de datos.

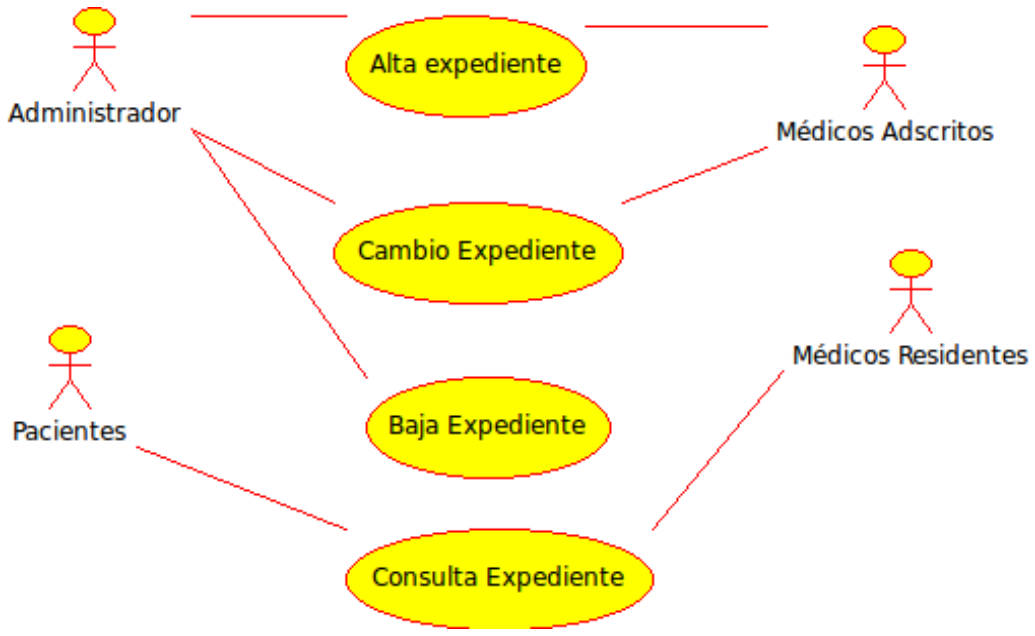
Autor:	Roberto González
Caso de Uso:	Cambio Examen
Descripción:	Modelar los escenarios que contempla la modificación de exámenes en el sistema.
Actores:	Administrador. Administrador que desee modificar un examen en el sistema. Médico Adscrito. Médico Adscrito que desee modificar un examen en el sistema.
Precondiciones:	Que exista el examen a modificar en la base de datos.
Escenario Principal:	El Administrador, Médico Adscrito, ingresa a la forma de modificación de exámenes. El Administrador, Médico Adscrito, llena la forma de modificación de exámenes.
Escenario Alternativo:	ninguno
Poscondiciones:	El examen se modifica en la base de datos.

Autor:	Roberto González
Caso de Uso:	Consulta Examen
Descripción:	Modelar los escenarios que contempla la consulta de exámenes en el sistema.
Actores:	Administrador. Administrador que desee consultar un examen en el sistema. Médico Adscrito. Médico Adscrito que desee consultar un examen en el sistema.
Precondiciones:	Que exista el examen a consultar en la base de datos.
Escenario Principal:	El Administrador, Médico Adscrito, ve una lista de exámenes. El Administrador, Médico Adscrito, elige un examen de la lista de exámenes, para ver su detalle.
Escenario Alternativo:	ninguno

Poscondiciones:	El Administrador, Médico Adscrito, ve el detalle del examen que eligió.

Autor:	Roberto González
Caso de Uso:	Realización Examen
Descripción:	Modelar los escenarios que contempla la realización de exámenes en el sistema.
Actores:	Médico Residente. Médico Residente que desee realizar un examen del sistema.
Precondiciones:	Que exista el examen a realizar en la base de datos.
Escenario Principal:	El Médico Residente ve una lista de exámenes. El Médico Residente elige que examen realizar.
Escenario Alternativo:	ninguno
Poscondiciones:	El Médico Residente realiza el examen y obtiene una calificación.

7.1.5 Expedientes



Autor:	Roberto González
Caso de Uso:	Alta Expediente
Descripción:	Modelar los escenarios que contempla la inserción de expedientes en el sistema.
Actores:	Administrador. Administrador que desee insertar un expediente en el sistema. Médico Adscrito. Médico Adscrito que desee insertar un expediente en el sistema.
Precondiciones:	Que se haya logueado correctamente el Administrador o el Médico Adscrito.
Escenario Principal:	El Administrador, Médico Adscrito, ingresa a la forma de alta de expedientes. El Administrador, Médico Adscrito, llena la forma de alta de expedientes.
Escenario Alternativo:	ninguno
Poscondiciones:	El expediente se inserta en la base de datos.

Autor:	Roberto González
Caso de Uso:	Cambio Expediente
Descripción:	Modelar los escenarios que contempla la modificación de expedientes en el sistema.
Actores:	Administrador. Administrador que desee modificar un expediente en el sistema. Médico Adscrito. Médico Adscrito que desee modificar un expediente en el sistema.
Precondiciones:	Que exista el expediente a modificar en la base de datos.
Escenario Principal:	El Administrador, Médico Adscrito, ingresa a la forma de cambio de expedientes. El Administrador, Médico Adscrito, llena la forma de cambio de expedientes.
Escenario Alternativo:	ninguno
Poscondiciones:	El expediente se modifica en la base de datos.

Autor:	Roberto González
Caso de Uso:	Baja Expediente
Descripción:	Modelar los escenarios que contempla la eliminación de expedientes en el sistema.
Actores:	Administrador. Administrador que desee eliminar un expediente en el sistema. Médico Adscrito. Médico Adscrito que desee eliminar un expediente en el

	sistema.
Precondiciones:	Que exista el expediente a eliminar en la base de datos.
Escenario Principal:	El Administrador, Médico Adscrito, ve una lista de expedientes. El Administrador, Médico Adscrito, elige el expediente a eliminar..
Escenario Alternativo:	ninguno
Poscondiciones:	El expediente se elimina de la base de datos.

Autor:	Roberto González
Caso de Uso:	Consulta Expediente
Descripción:	Modelar los escenarios que contempla la consulta de expedientes en el sistema.
Actores:	Administrador. Administrador que desee consultar un expediente en el sistema. Médico Adscrito. Médico Adscrito que desee consultar un expediente en el sistema. Médico Residente. Médico Residente que desee consultar un expediente en el sistema Paciente. Paciente que desee consultar un expediente en el sistema.
Precondiciones:	Que exista el expediente a consultar en la base de datos.
Escenario Principal:	El Administrador, Médico Adscrito, Médico Residente y Paciente ve un listado de expedientes. El Administrador, Médico Adscrito, Médico Residente y Paciente elige un expediente a consultar..
Escenario Alternativo:	ninguno
Poscondiciones:	El Administrador, Médico Adscrito, Médico Residente y Paciente ve el detalle del expediente que eligió.

7.2 Diagrama Entidad-Relación

Se identificaron las tablas a utilizar en el sistema, este diagrama contiene los campos de cada tabla, su tipo, así como las relaciones que guardan las tablas entre sí. El Diagrama Entidad-Relación se puede consultar en la carpeta adjunta a este reporte, por cuestiones de espacio no se puede mostrar aquí, en la carpeta Base de Datos, se encuentra dicho diagrama, así como los scripts para la creación de la Base de Datos.

7.2.1 Diccionario de Datos de la Base de Datos

Tabla:

usuarios = tabla que contendrá los campos de de los usuarios, se relaciona con las tablas medicos_residentes, medicos_adscritos, administradores y pacientes

Campos:

id = campo que contendrá el id del usuario, bigint, sin signo, no puede ser nulo, llave primaria de la tabla usuarios, se auto-incrementa en uno conforme se insertar un usuario nuevo.

login = campo que contendrá el login del usuario, varchar, no puede ser nulo.

password = campo que contendrá el password del usuario, varchar, no puede ser nulo.

nombres = campo que contendrá los nombres del usuario, varchar, no puede ser nulo.

ap_paterno = campo que contendrá el apellido paterno del usuario, varchar, no puede ser nulo.

ap_materno = campo que contendrá el apellido materno del usuario, varchar, puede ser nulo.

tel_casa = campo que contendrá el teléfono de casa del usuario, varchar, no puede ser nulo.

tel_trabajo = campo que contendrá el teléfono de trabajo del usuario, varchar, puede ser nulo.

fax = campo que contendrá el fax del usuario, varchar, puede ser nulo.

celular = campo que contendrá el celular del usuario, varchar, puede ser nulo.

direccion = campo que contendrá la dirección del usuario, varchar, no puede ser nulo.

direccion_2 = campo que contendrá la dirección 2 del usuario, varchar, puede ser nulo.

codigo_postal = campo que contendrá el código postal del usuario, varchar, no puede ser nulo

email = campo que contendrá el email del usuario, varchar, no puede ser nulo.

genero = campo que contendrá el género del usuario, char, no puede ser nulo.

fecha_nacimiento = campo que contendrá la fecha de nacimiento del usuario, Date, no puede ser nulo.

fecha_ingreso = campo que contendrá la fecha de ingreso al sistema del usuario, Date, no puede ser nulo.

pais = campo que contendrá el de país del usuario, varchar, no puede ser nulo.

iperfil = campo que contendrá el id del perfil al que pertenece el usuario, int, sin signo, no puede ser nulo.

status = campo que contendrá el estatus del usuario, int, no puede ser nulo, 0 significa que está habilitado el usuario y 1 que está deshabilitado.

Tabla:

administrador = tabla que contendrá los campos del administrador, se relaciona con la tabla usuarios.

Campos:

id = campo que contendrá el id del administrador, bigint, sin signo, no puede ser nulo, se auto-incrementa cuando se inserte un nuevo registro en la tabla, se relaciona con la tabla usuario, esto para cumplir con la relación de herencia entre usuario y administrador.

email_alternativo = campo que contendrá el email del administrador, varchar, puede ser nulo.

Tabla:

casos: tabla que contendrá los campos de los casos.

Campos:

id = campo que contendrá que el id del caso, bigint, sin signo, no puede ser nulo, es la llave primaria de la tabla casos, se auto-incrementa conforme se agreguen nuevos registros a la tabla.

titulo = campo que contendrá el titulo del caso, varchar, no puede ser nulo.

contenido_caso = campo que contendrá el contenido del caso, varchar, no puede ser nulo.

status = campo que contendrá el estatus del caso, int, no puede ser nulo.

Tabla:

contenidos = tabla que contendrá los campos de los contenidos, se relaciona con las tablas, referencias, subcontenidos y temas_relacionados.

Campos:

id_contenidos = campo que contendrá el id del contenido, bigint, sin signo, no puede ser nulo, es la llave primaria de la tabla contenidos, se auto-incrementa conforme se agreguen nuevos registros a la tabla.

titulo_contenido = campo que contendrá el título del contenido, varchar, no puede ser nulo.

descripcion = campo que contendrá la descripción del contenido, varchar, no puede ser nulo.

status = campo que contendrá el status del contenido, int, no puede ser nulo.

Tabla:

examenes = tabla que contendrá los campos de los exámenes, se relaciona con las tablas de medicos_residentes en una relación muchos a muchos, usando como tabla intermedia a la tabla medicos_residentes_examenes, y también con la tabla preguntas.

Campos:

id_examenes = campo que contendrá el id del examen, bigint, sin signo, no puede ser nulo, es la llave primaria de la tabla examenes, se auto-incrementa conforme se agreguen nuevos registros a la tabla.

titulo = campo que contendrá el titulo del examen, varchar, no puede ser nulo.

status = campo que contendrá el estatus del examen, int, no puede ser nulo.

Tabla:

medicos_adscritos = tabla que contendrá los campos de los médicos adscritos, se relaciona con la tabla usuarios, mediante el campo id.

Campos:

id = campo que contendrá el id del médico adscrito, bigint, sin signo, no puede ser nulo, llave primaria de la tabla medicos_adscritos, se auto-incrementa conforme se agreguen nuevos registros a la tabla.

area = campo que contendrá el área a la que pertenece el médico adscrito, puede ser nulo.

puesto = campo que contendrá el puesto del médico adscrito, puede ser nulo.

Tabla:

medicos_residentes = tabla que contendrá los campos de los médicos residentes, se relaciona con la tabla usuarios, mediante el campo id.

Campos:

id = campo que contendrá del médico residente, bigint, sin signo, no puede ser nulo, es la llave primaria de la tabla medicos_residentes, se auto-incrementa conforme se agreguen registros a la tabla.

escuela = campo que contendrá la escuela del médico residente, varchar, no puede ser nulo.

Tabla:

pacientes = tabla que contendrá los campos de los pacientes, se relaciona con la tabla usuarios, mediante el campo id.

Campos:

id = campo que contendrá del paciente, bigint, sin signo, no puede ser nulo, es la llave primaria de la tabla pacientes, se auto-incrementa conforme se agreguen registros a la tabla.

no_afiliacion = campo que contendrá el número de afiliación del paciente, varchar, no puede ser nulo.

tipo_sangre = campo que contendrá el tipo de sangre del paciente, varchar, no puede ser nulo.

alergias = campo que contendrá las alergias del paciente, varchar, puede ser nulo.

cuidados = campo que contendrá los cuidados del paciente, varchar, puede ser nulo.

enfermedades = campo que contendrá las enfermedades del paciente, varchar, puede ser nulo.
Tabla:

preguntas = tabla que contendrá las preguntas del sistema, se relaciona con la tabla examenes mediante el campo id_examenes, y con la tabla respuestas, mediante el campo id_preguntas.

Campos:

id_preguntas = campo que contendrá el id de la pregunta, bigint, sin signo, no puede ser nulo, se auto-incrementa conforme se agreguen registros a la tabla, es la llave primaria de la tabla preguntas.

Tabla:

referencias = tabla que contendrá las referencias de los contenidos, se relaciona con la tabla contenidos, mediante el campo id_contenidos.

Campos:

id_referencias = campo que contendrá el id de la referencia, bigint, sin signo, no puede ser nulo, se auto-incrementa conforme se agreguen registros a la tabla, es la llave primaria de la tabla referencias.

id_contenidos = campo que contendrá el id del contenido con el cual está relacionada la referencia, se relaciona con la tabla contenidos, bigint, puede ser nulo.

referencia = campo que contendrá la referencia, varchar, no puede ser nulo.

link = campo que contendrá el hipervínculo de la referencia, varchar, puede ser nulo.

status = campo que contendrá el estatus de la referencia, int, no puede ser nulo.

Tabla:

respuestas = tabla que contendrá las respuesta de las preguntas, se relaciona con la tabla preguntas, mediante el campo id_preguntas;

Campos:

id_respuestas = campo que contendrá el id de la respuesta, bigint, sin signo, no puede ser nulo, se auto-incrementa conforme se agreguen registros a la tabla, es la llave primaria de la tabla respuestas.

id_preguntas = campo que contendrá el id de la pregunta con la cual está relacionada la respuesta, se relaciona con la tabla preguntas, bigint, sin signo, puede ser nulo.

respuesta = campo que contendrá la respuesta, varchar, no puede ser nulo.

correcta = campo que indicará si la respuesta es correcta, int, no puede ser nulo. 0 para respuesta correcta y 1 para respuesta incorrecta.

status = campo que contendrá el estatus de la respuesta, int, no puede ser nulo.

Tabla:

subcontenidos = tabla que contendrá los subcontenidos relacionados con los subcontenidos, se relaciona con la tabla contenidos, mediante el campo id_contenidos.

Campos:

id_subcontenidos = campo que contendrá el id del subcontenido, bigint, sin signo, no puede ser nulo, se auto-incrementa conforme se agreguen registros a la tabla, es la llave primaria de la tabla subcontenidos.

id_contenidos = campo que contendrá el id del contenido con el cual está relacionado el subcontenido, se relaciona con la tabla contenidos, bigint, sin signo, puede ser nulo.

descripcion_subcontenido = campo que contendrá la descripción del subcontenido, varchar, no puede ser nulo.

status = campo que contendrá el estatus del subcontenido, int, no puede ser nulo.

Tabla:

temas_relacionados = tabla que contendrá los temas relacionados asociados a un contenido, se relaciona con la tabla contenidos, mediante el campo id_contenidos.

Campos:

id_temas_relacionado = campo que contendrá el id del tema relacionado, bigint, sin signo, no puede ser nulo, auto-incrementa conforme se agreguen registros a la tabla, es la llave primaria de la tabla temas_relacionados.

id_contenidos = campo que contendrá el id del contenido al cual está relacionado el tema relacionado, se relaciona con la tabla contenido, bigint, sin signo, puede ser nulo.

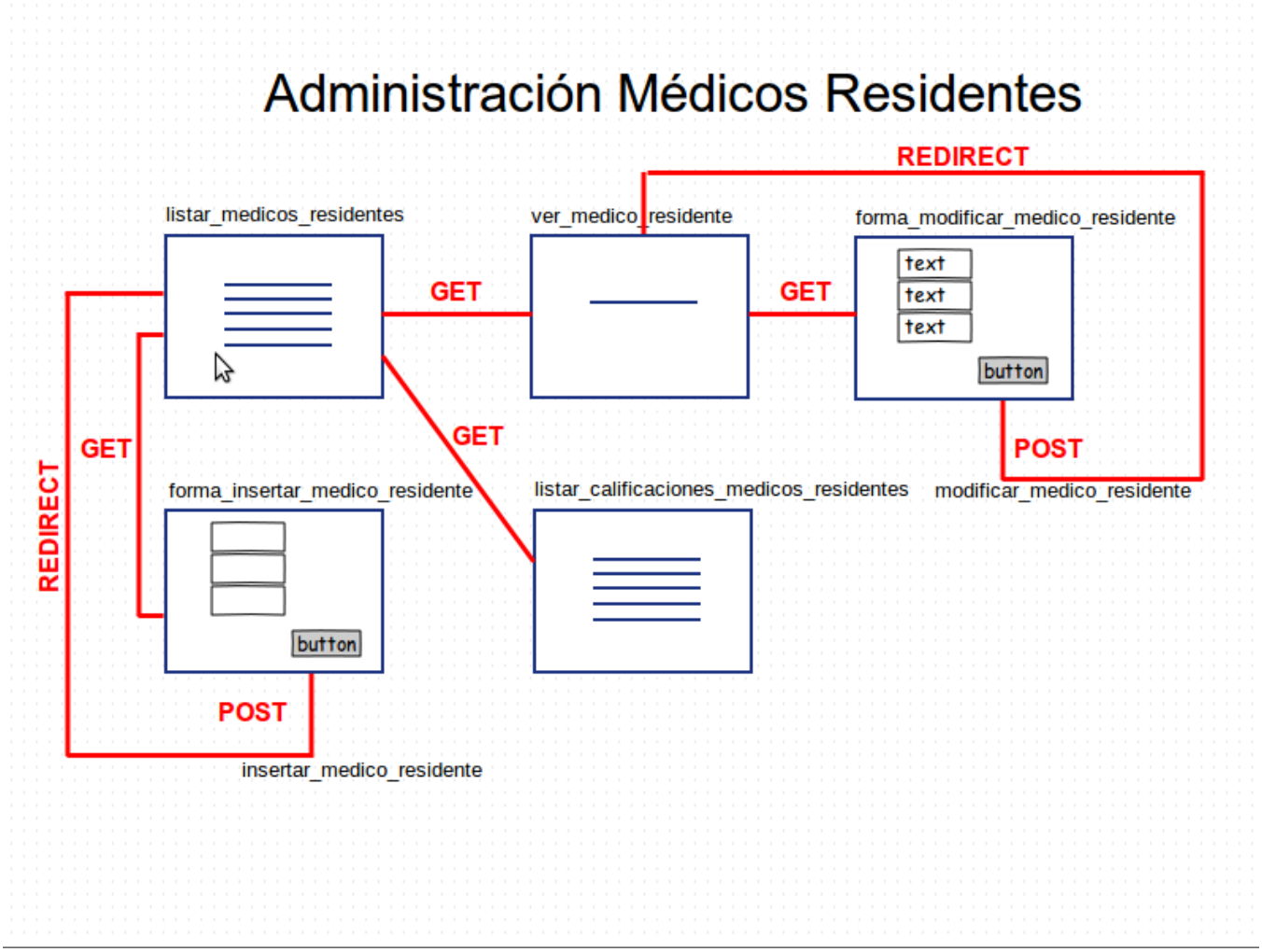
tema_relacionado = campo que contendrá el tema relacionado, varchar, no puede ser nulo.

link = campo que contendrá el link del tema relacionado, varchar, puede ser nulo.

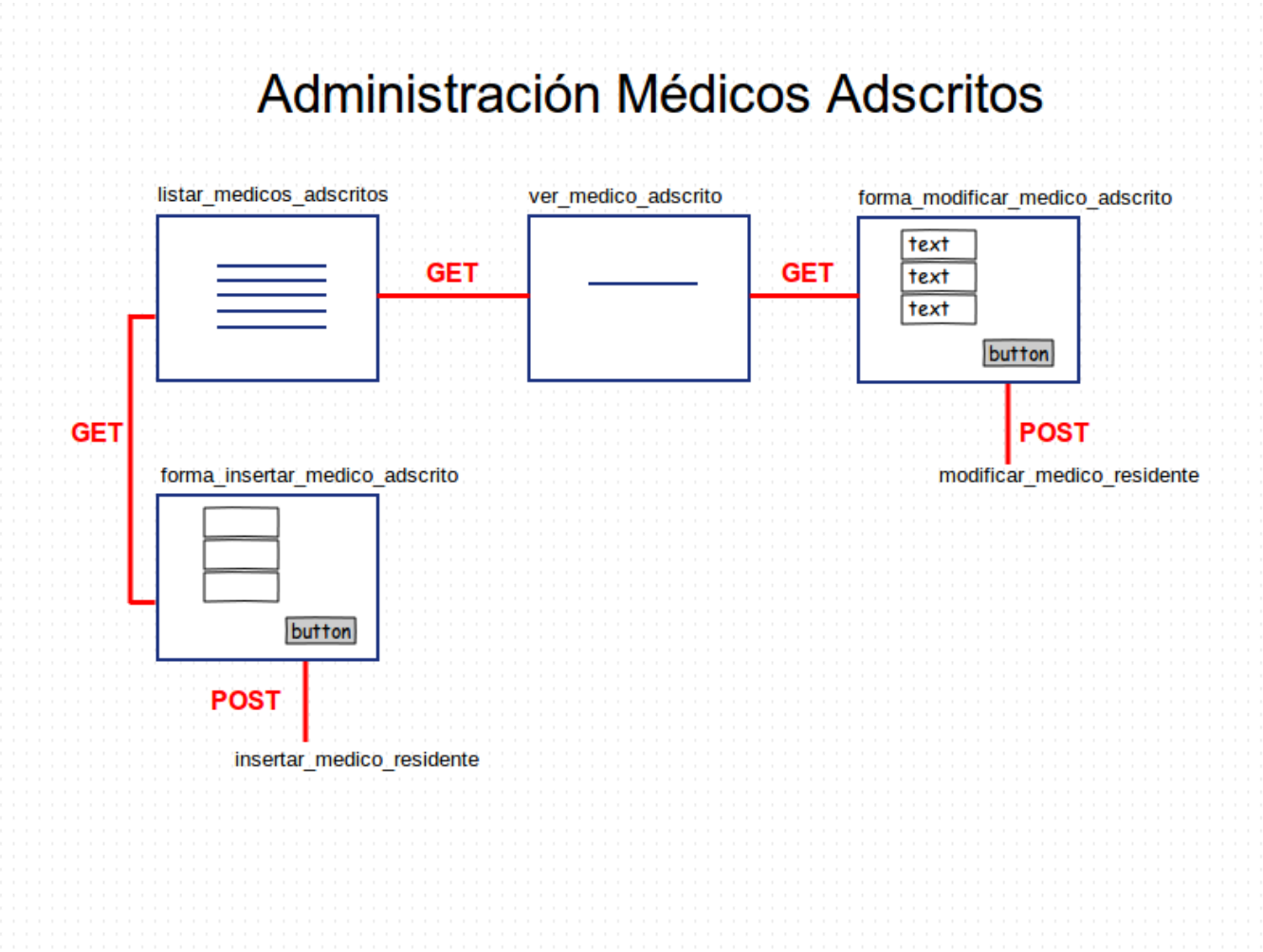
status = campo que contendrá el estatus del tema relacionado, no puede ser nulo.

7.3 Diagrama de Navegación entre páginas y clases

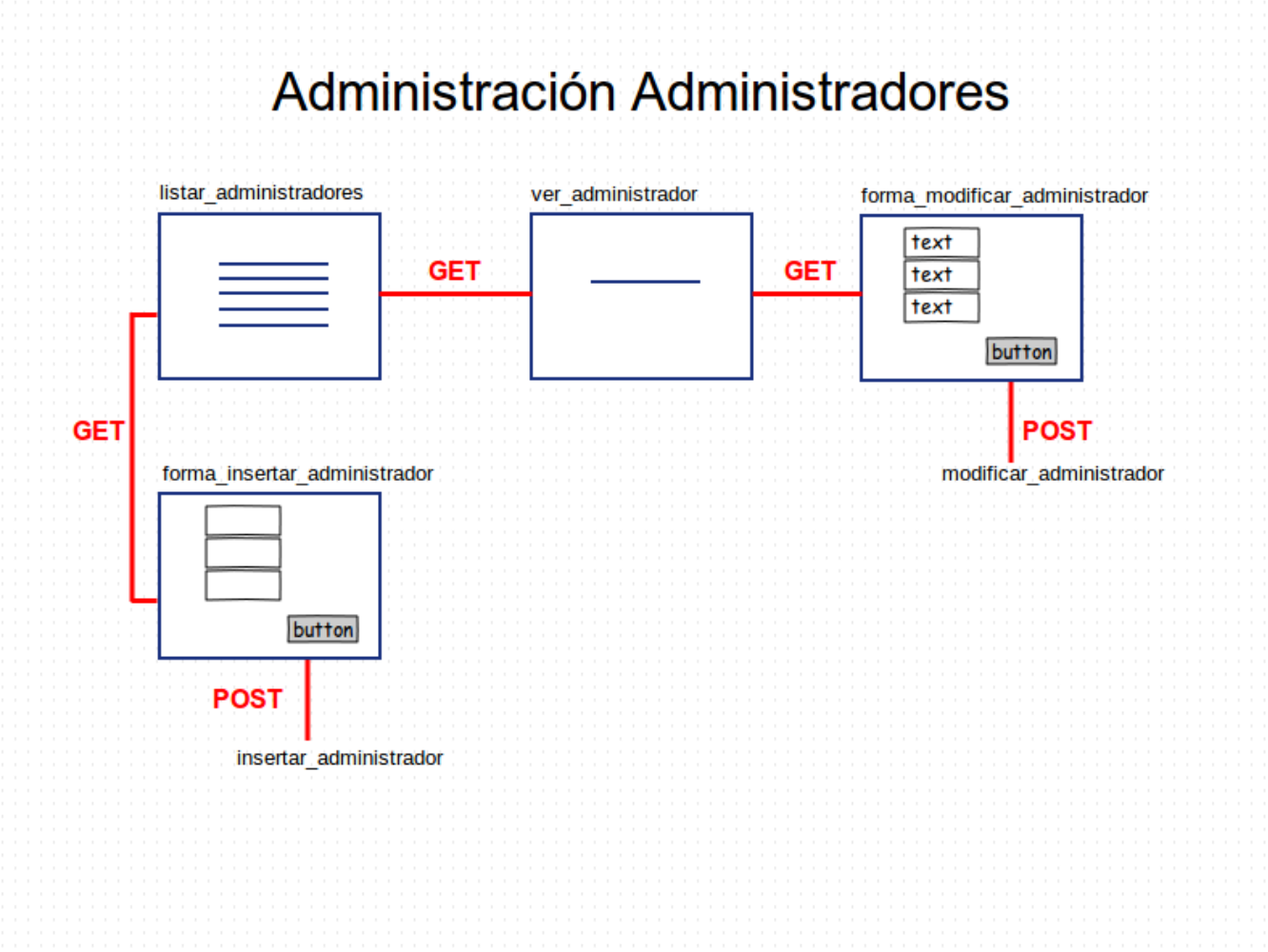
7.3.1 Médicos Residentes


Administración Médicos Residentes
REDIRECT
listar_medicos_residentes
ver_medico_residente
forma_modificar_medico_residente
GET
GET
text
text
text
button
GET
GET
POST
REDIRECT
forma_insertar_medico_residente
listar_calificaciones_medicos_residentes
modificar_medico_residente
button
POST
insertar_medico_residente

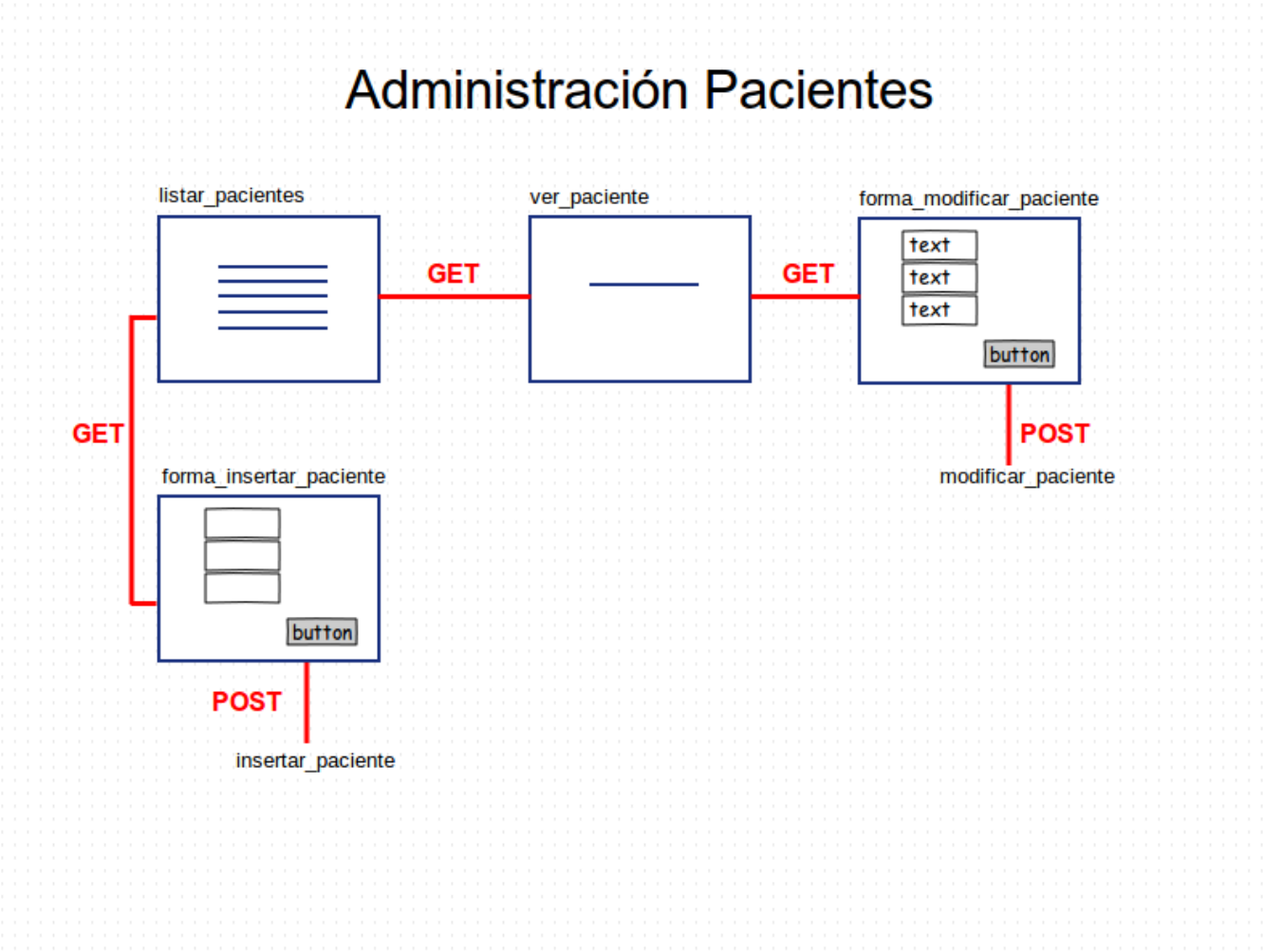
7.3.2 Médicos Adscritos



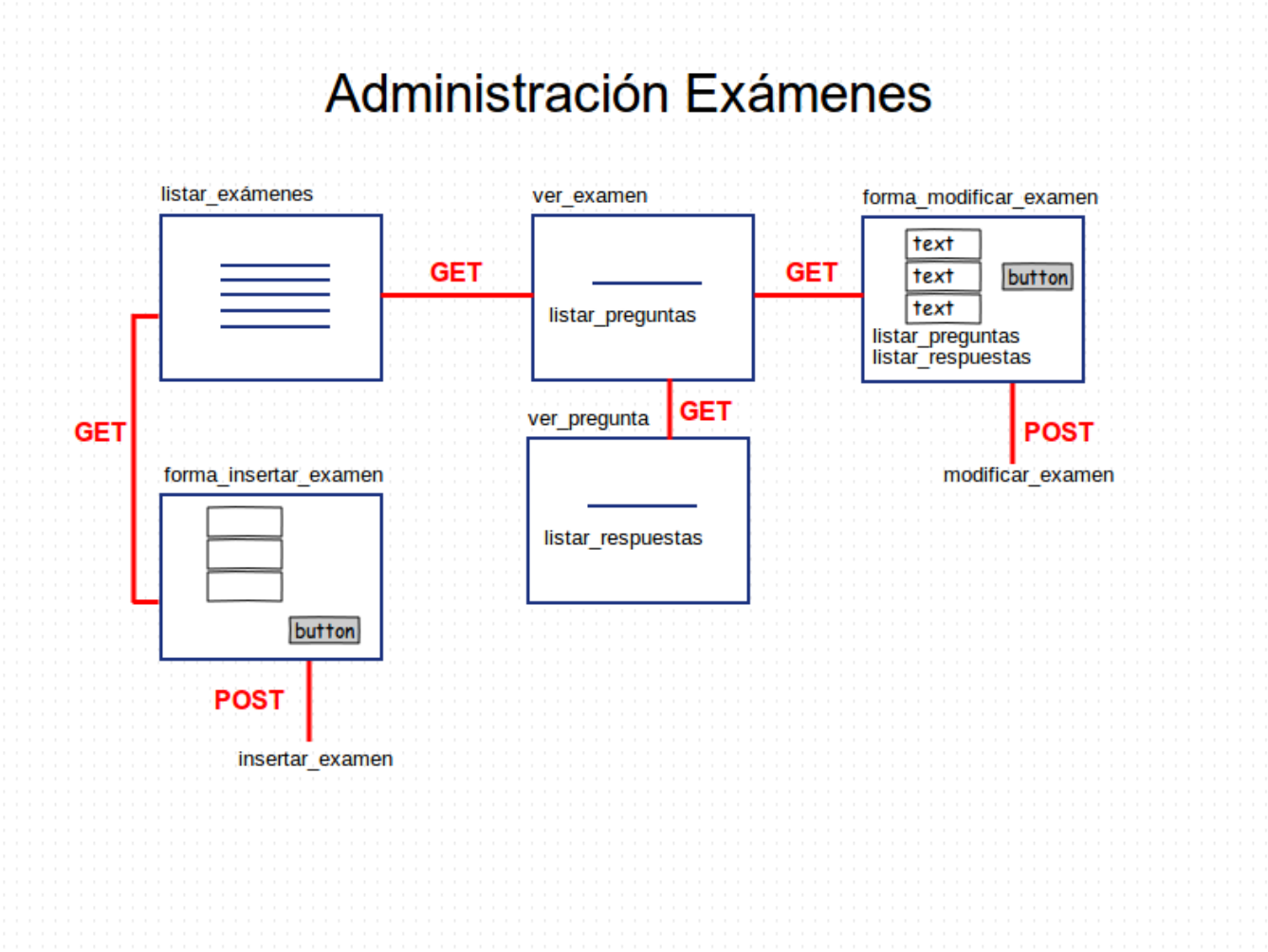
7.3.3 Administradores



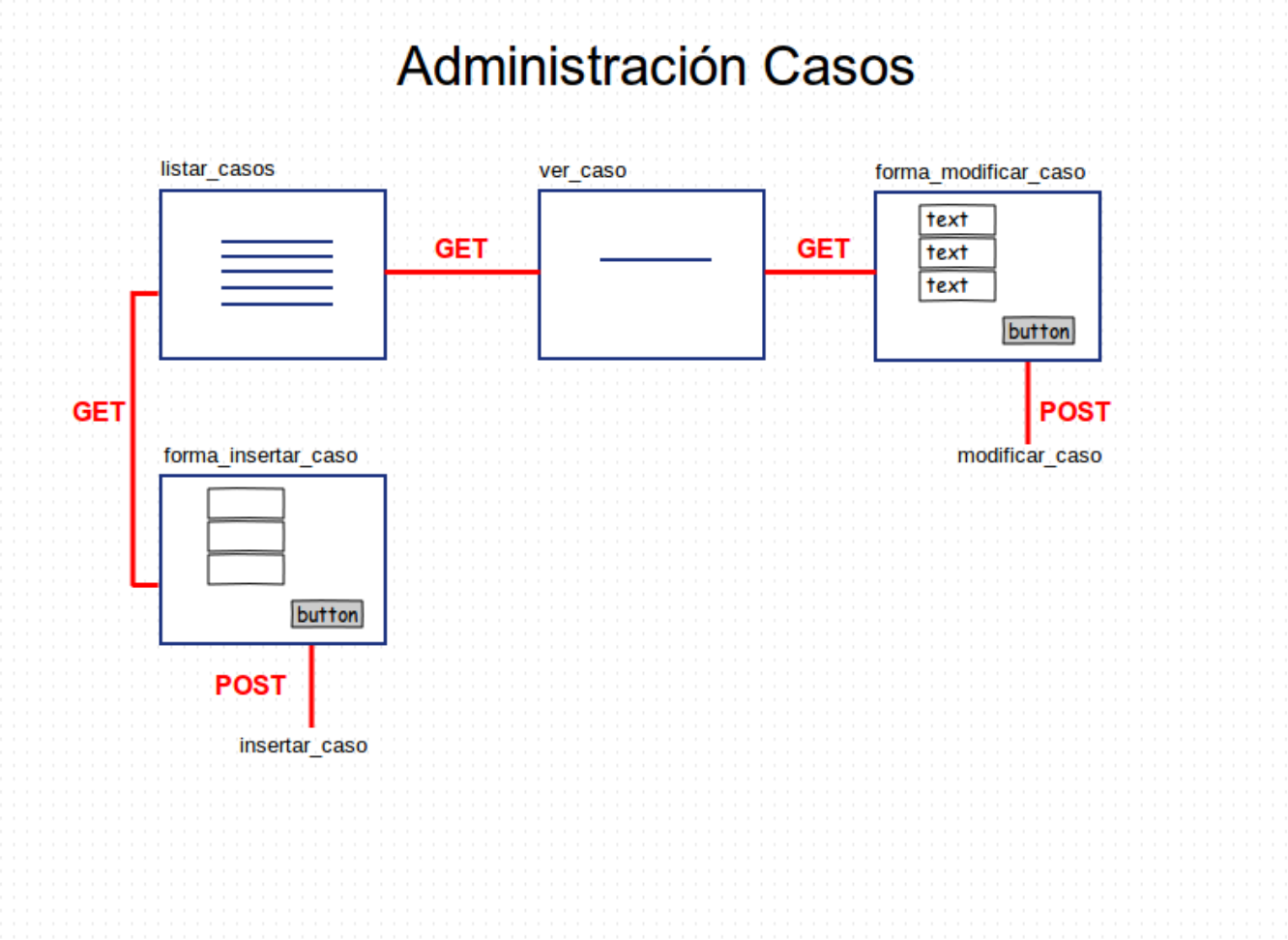
7.3.4 Pacientes



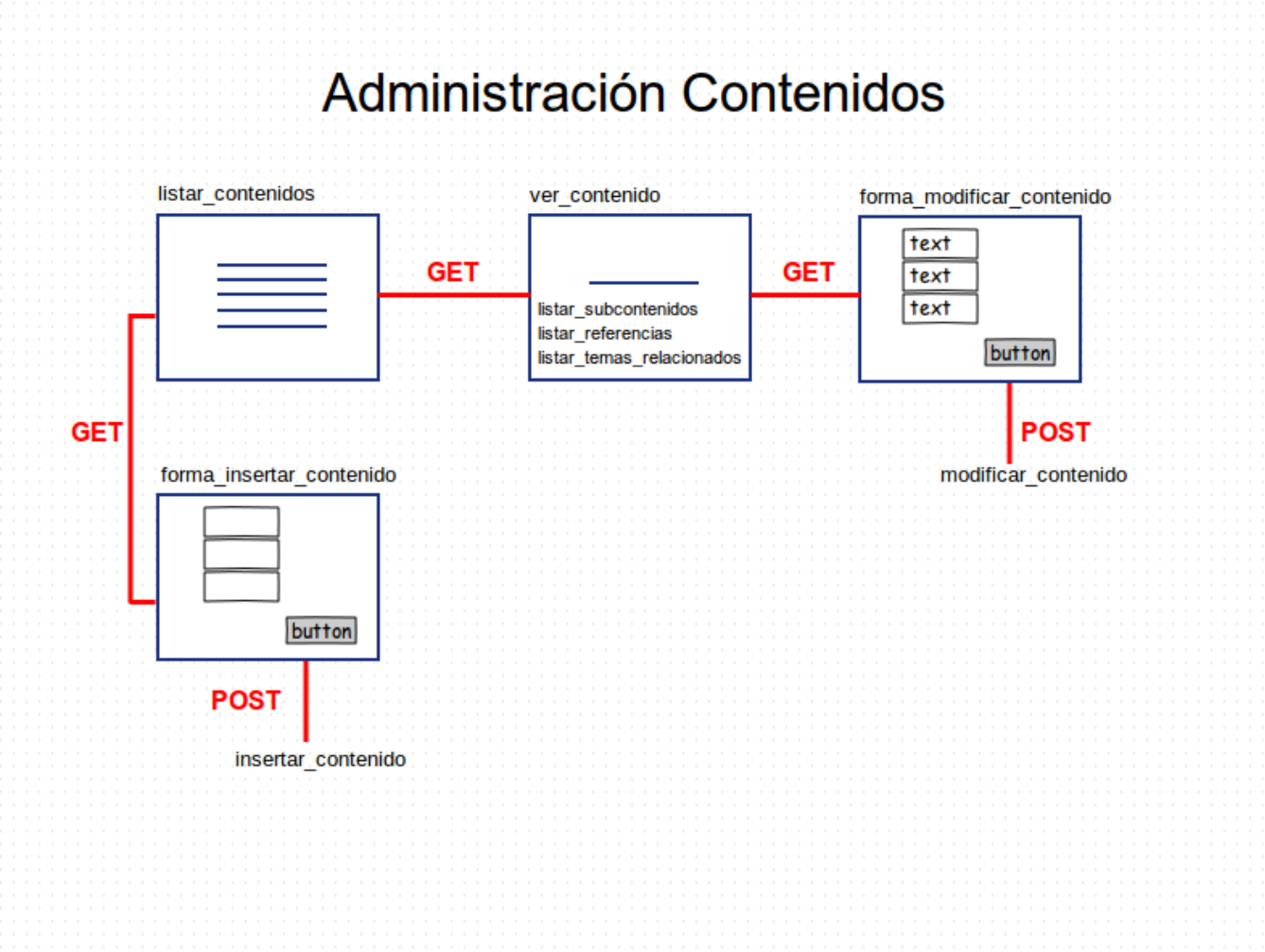
7.3.5 Exámenes



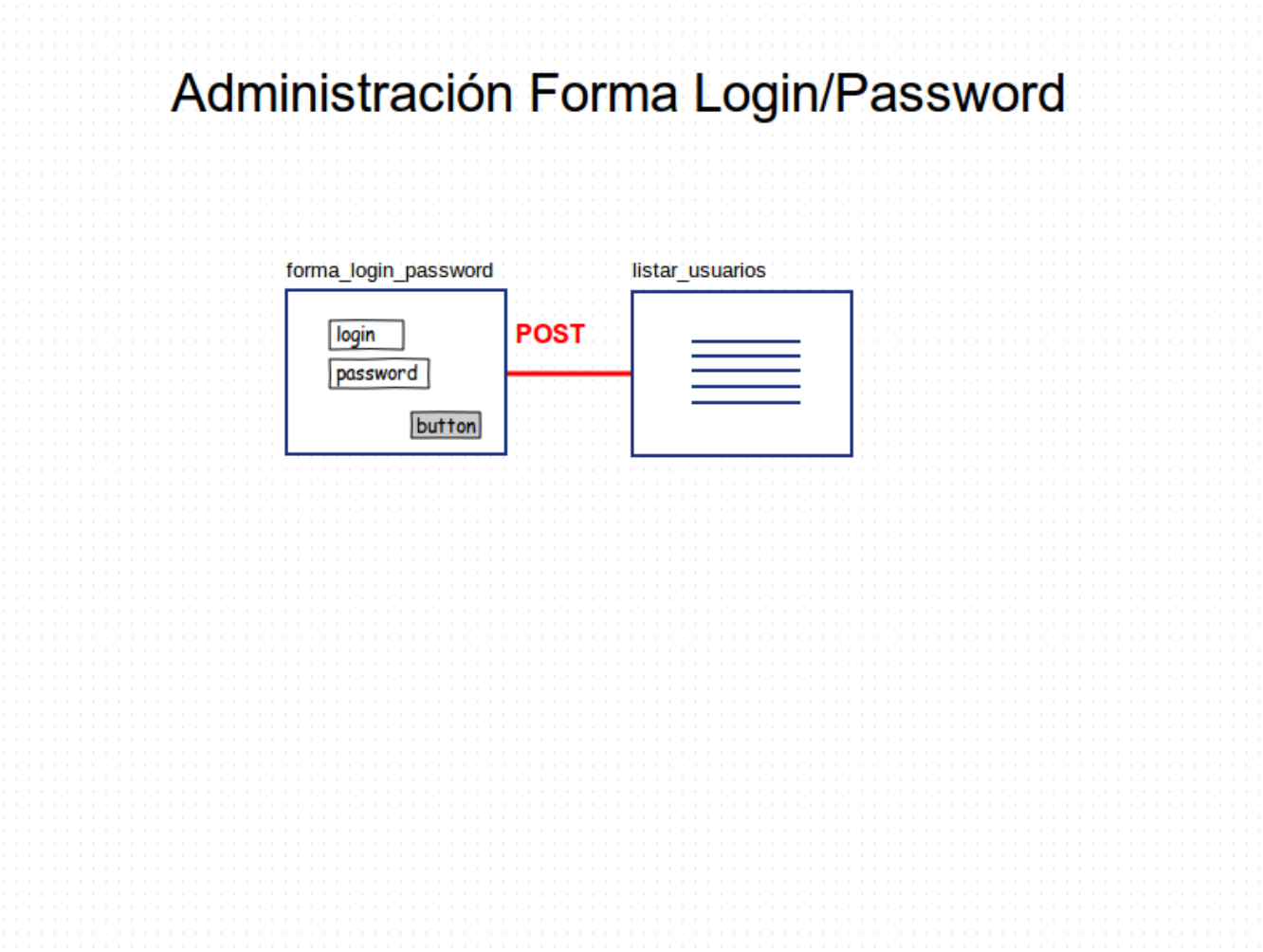
7.3.6 Casos


Administración Casos
listar_casos
ver_caso
forma_modificar_caso
GET
GET
text
text
text
button
GET
POST
forma_insertar_caso
modificar_caso
button
POST
insertar_caso

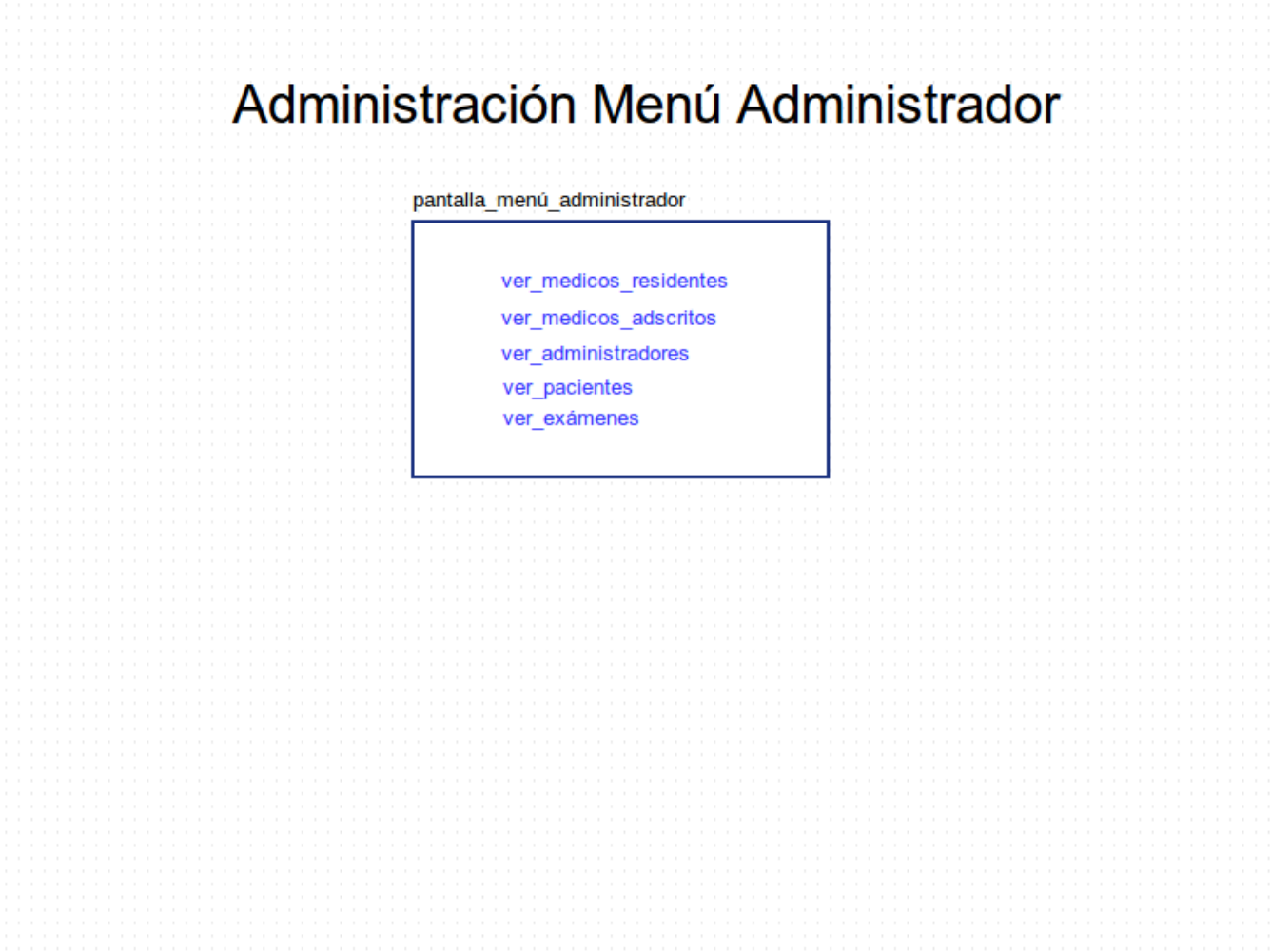
7.3.7 Contenidos



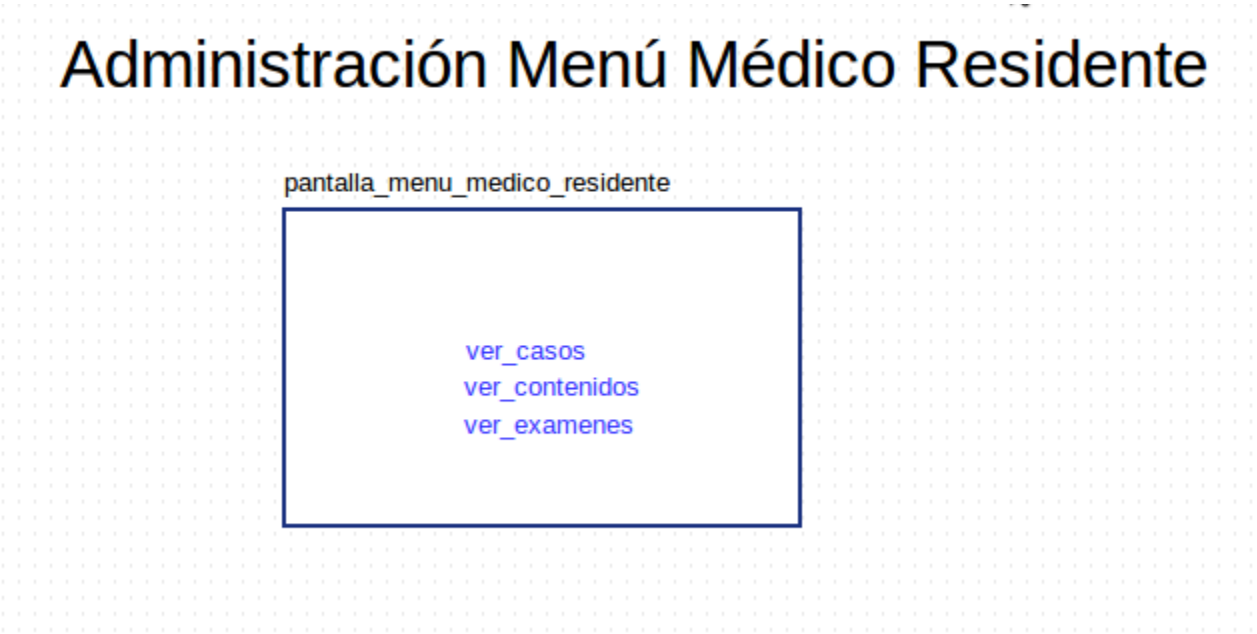
7.3.8 login


Administración Forma Login/Password
forma_login_password
listar_usuarios
login
password
button
POST

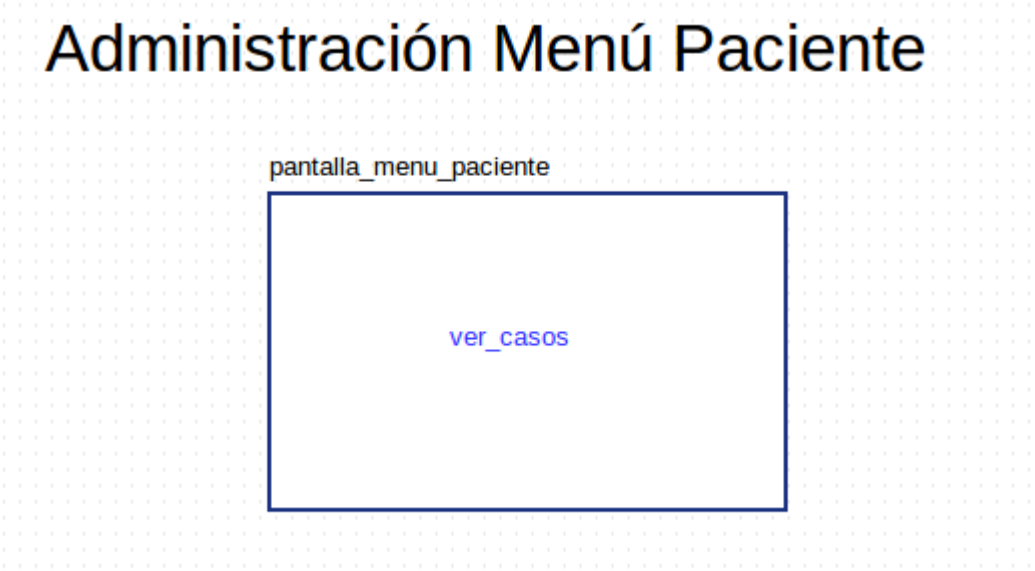
7.3.9 Menú Administrador



7.3.10 Menú Médicos Residentes



7.3.11 Menú Pacientes



7.4 Diagrama de Clases

El diagrama de clases, en el cual se muestran los métodos públicos de cada clase, así como la interacción entre ellas, por cuestiones de espacio se encuentra en la carpeta diagramas.

7.5 Manual de Usuario

El manual, en el cual se muestra el funcionamiento del sistema, así como la interacción entre el usuario y el sistema, por cuestiones de espacio se encuentra en la carpeta manual_usuario.

8 Código Fuente

8.1 mx.uam.azc.pt.business

8.1.1 Administrador Administradores

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.business;

import java.util.List;

import mx.uam.azc.pt.business.local.FiltroAdministradores;
import mx.uam.azc.pt.business.local.FiltroMedicosAdscritos;
import mx.uam.azc.pt.data.AdministradorJoinDTO;
import mx.uam.azc.pt.data.MedicoAdscritoJoinDTO;

import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

/**
 * Interfase que provee de los métodos para la administración de Administradores
```

```
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Transactional(rollbackFor=Exception.class)
public interface AdministradorAdministradores {

	/**
	* Inserta un administrador al sistema
	* @param administrador Datos del administrador
	*/
	public void insertarAdministrador(AdministradorJoinDTO administrador);

	/**
	   * Lee los datos asociados a un administrador
	   * @param id Identificador del administrador
	   * @return El DTO con los datos del administrador
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
	public AdministradorJoinDTO leerAdministrador(long id);

	/**
	   * Actualiza un administrador en el sistema
	   * @param administrador Datos del administrador
	   */
	public void actualizarAdministrador(AdministradorJoinDTO administrador);

	/**
	   * Lee una lista de administradores
	   * @return La lista de DTOs con la información de los administradores
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
	public List<AdministradorJoinDTO> leerAdministradores();

	/**
	 * Metodo que lee una lista de administradores de acuerdo a un filtro
proporcionado
	 * @param filter filtro que fungira como patron de busqueda
	 * @return La lista de administradores que cumplieron el patron de busqueda
	 */
	@Transactional( propagation=Propagation.SUPPORTS, readOnly = true )
	public List<AdministradorJoinDTO>
leerAdministradoresFiltro( FiltroAdministradores filter );

}
```

8.1.2 Administrador Casos

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
```

```java
package mx.uam.azc.pt.business;

import java.util.List;

import mx.uam.azc.pt.business.local.FiltroCasos;
import mx.uam.azc.pt.data.CasoJoinDTO;

import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

/**
 * Interfase que provee de los métodos para la administración de Casos
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Transactional( rollbackFor=Exception.class )
public interface AdministradorCasos {

	/**
	* Inserta un caso al sistema
	* @param caso Datos del Caso
	*/
	public void insertarCaso(CasoJoinDTO caso);

	/**
	   * Actualiza un caso en el sistema
	   * @param caso Datos del Caso
	   */
	public void actualizarCaso(CasoJoinDTO caso);

	/**
	   * Lee una lista de casos
	   * @return La lista de DTOs con la información de los casos
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
	public List<CasoJoinDTO> leerCasos();

	/**
	   * Lee los datos asociados a un caso
	   * @param id Identificador del caso
	   * @return El DTO con los datos del caso
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
	public CasoJoinDTO leerCaso(long id);

	/**
	 * Metodo que Lee una lista de casos de acuerdo a nu patron de busqueda
	 * @param filter Filtro que fungira como patron de busqueda
	 * @return La lista de casos que cumplieron con el patron de busqueda
	 */
	@Transactional( propagation=Propagation.SUPPORTS, readOnly = true )
	public List<CasoJoinDTO> leerCasosFiltro( FiltroCasos filter );

}
```

8.1.3 AdministradorContenidos

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLÁSICAS
 */
package mx.uam.azc.pt.business;

import java.util.List;

import mx.uam.azc.pt.business.local.FiltroCasos;
import mx.uam.azc.pt.business.local.FiltroContenidos;
import mx.uam.azc.pt.data.CasoJoinDTO;
import mx.uam.azc.pt.data.ContenidoJoinDTO;
import mx.uam.azc.pt.data.ReferenciaDTO;
import mx.uam.azc.pt.data.SubContenidoDTO;
import mx.uam.azc.pt.data.TemaRelacionadoJoinDTO;

import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

/**
 * Interfase que provee de los métodos para la administración de Contenidos
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Transactional( rollbackFor=Exception.class )
public interface AdministradorContenidos {

	/**
	   * Lee los datos asociados a un contenido
	   * @param id Identificador del contenido
	   * @return El DTO con los datos del contenido
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
	public ContenidoJoinDTO leerContenido(long id);

	/**
	   * Lee los datos asociados a un subcontenido
	   * @param id Identificador del subcontenido
	   * @return El DTO con los datos del subcontenido
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
	public SubContenidoDTO leerSubContenido(long id);

	/**
	   * Lee los datos asociados a una referencia
	   * @param id Identificador de la referencia
	   * @return El DTO con los datos de la referencia
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
```

```
public ReferenciaDTO leerReferencia(long id);

/**
   * Lee los datos asociados a un tema relacionado
   * @param id Identificador del tema relacionado
   * @return El DTO con los datos del tema relacionado
   */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public TemaRelacionadoJoinDTO leerTemaRelacionado(long id);

/**
   * Lee una lista de contenidos
   * @return La lista de DTOs con la información de los contenidos
   */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public List<ContenidoJoinDTO> leerContenidos();

/**
   * Lee una lista de subcontenidos
   * @return La lista de DTOs con la información de los subcontenidos
   */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public List<SubContenidoDTO> leerSubContenidos();

/**
   * Lee una lista de referencias
   * @return La lista de DTOs con la información de las referencias
   */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public List<ReferenciaDTO> leerReferencias();

/**
   * Lee una lista de temas relacionados
   * @return La lista de DTOs con la información de los temas relacionados
   */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public List<TemaRelacionadoJoinDTO> leerTemasRelacionados();

/**
* Inserta un contenido al sistema
* @param contenido Datos del contenido
*/
public void insertarContenido(ContenidoJoinDTO contenido);

/**
* Inserta un subcontenido al sistema
* @param subcontenido Datos del subcontenido
*/
public void insertarSubContenido(SubContenidoDTO subcontenido);

/**
* Inserta una referencia al sistema
* @param referencia Datos de la referencia
*/
```

```
	public void insertarReferencia(ReferenciaDTO referencia);

	/**
	* Inserta un tema relacionado al sistema
	* @param tema_relacionado Datos del tema relacionado
	*/
	public void insertarTemaRelacionado(TemaRelacionadoJoinDTO
tema_relacionado);

	/**
	   * Actualiza un contenido en el sistema
	   * @param contenido Datos del contenido
	   */
	public void actualizarContenido(ContenidoJoinDTO contenido);

	/**
	   * Actualiza un subcontenido en el sistema
	   * @param subcontenido Datos del subcontenido
	   */
	public void actualizarSubContenido(SubContenidoDTO subcontenido);

	/**
	   * Actualiza una refrencia en el sistema
	   * @param referencia Datos de la referencia
	   */
	public void actualizarReferencia(ReferenciaDTO referencia);

	/**
	   * Actualiza un tema relacionado en el sistema
	   * @param tema_relacionado Datos del tema relacionado
	   */
	public void actualizarTemaRelacionado(TemaRelacionadoJoinDTO
tema_relacionado);

	/**
	 *
	 * Metodo que lee una lista de contenidos de acunerdo filtro proporcionado
	 * @param filter Filtro que fungira como patron de busqueda
	 * @return La lista de contenidos que cumplieron el filtro de busqueda
	 */
	@Transactional( propagation=Propagation.SUPPORTS, readOnly = true )
	public List<ContenidoJoinDTO> leerContenidosFiltro( FiltroContenidos
filter );

}
```

8.1.4 AdministradorExamenes

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
```

```
 */
package mx.uam.azc.pt.business;

import java.util.List;

import mx.uam.azc.pt.business.local.FiltroContenidos;
import mx.uam.azc.pt.business.local.FiltroExamenes;
import mx.uam.azc.pt.data.ContenidoJoinDTO;
import mx.uam.azc.pt.data.ExamenJoinDTO;
import mx.uam.azc.pt.data.PreguntaDTO;
import mx.uam.azc.pt.data.RespuestaDTO;

import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

/**
 * Interfase que provee de los métodos para la administración de Exámenes
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Transactional( rollbackFor=Exception.class )
public interface AdministradorExamenes {

	/**
	* Inserta un examen al sistema
	* @param examen Datos del examen
	*/
	public void insertarExamen(ExamenJoinDTO examen);

	/**
	* Inserta una pregunta al sistema
	* @param pregunta Datos de la pregunta
	*/
	public void insertarPregunta(PreguntaDTO pregunta);

	/**
	* Inserta una respuesta al sistema
	* @param respuesta Datos de la respuesta
	*/
	public void insertarRespuesta(RespuestaDTO respuesta);

	/**
	   * Actualiza un examen en el sistema
	   * @param examen Datos del examen
	   */
	public void actualizarExamen(ExamenJoinDTO examen);

	/**
	   * Actualiza una pregunta en el sistema
	   * @param pregunta Datos de la pregunta
	   */
	public void actualizarPregunta(PreguntaDTO pregunta);

	/**
```

```
    * Actualiza una respuesta en el sistema
    * @param respuesta Datos de la respuesta
    */
public void actualizarRespuesta(RespuestaDTO respuesta);

/**
    * Lee los datos asociados a un examen
    * @param id Identificador del examen
    * @return El DTO con los datos del examen
    */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public ExamenJoinDTO leerExamen(long id);

/**
    * Lee los datos asociados a una pregunta
    * @param id Identificador de la pregunta
    * @return El DTO con los datos de la pregunta
    */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public PreguntaDTO leerPregunta(long id);

/**
    * Lee los datos asociados a una respuesta
    * @param id Identificador de la respuesta
    * @return El DTO con los datos de la respuesta
    */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public RespuestaDTO leerRespuesta(long id);

/**
    * Lee una lista de examenes
    * @return La lista de DTOs con la información de los examenes
    */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public List<ExamenJoinDTO> leerExamenes();

/**
    * Lee una lista de preguntas
    * @return La lista de DTOs con la información de las preguntas
    */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public List<PreguntaDTO> leerPreguntas();

/**
    * Lee una lista de respuestas
    * @return La lista de DTOs con la información de las respuestas
    */
@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
public List<RespuestaDTO> leerRespuestas();

/**
 * Metodo que lee una lista de examenes de acuerdo a unfiltro de busqueda
 * @param filter Fitro que fungira como patron de busqueda
 * @return
```

```
	 */
	@Transactional( propagation=Propagation.SUPPORTS, readOnly = true )
	public List<ExamenJoinDTO> leerExamenesFiltro( FiltroExamenes filter );

}
```

8.1.5 AdministradorMedicosAdscritos

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.business;

import java.util.List;

import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import mx.uam.azc.pt.business.local.FiltroMedicosAdscritos;
import mx.uam.azc.pt.business.local.FiltroMedicosResidentes;
import mx.uam.azc.pt.data.MedicoAdscritoJoinDTO;
import mx.uam.azc.pt.data.MedicoResidenteJoinDTO;

/**
 * Interfase que provee de los métodos para la administración de Médicos Adscritos
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Transactional( rollbackFor=Exception.class )
public interface AdministradorMedicosAdscritos {

	/**
	* Inserta un médico adscrito al sistema
	* @param medico_adscrito Datos del médico adscrito
	*/
	public void insertarMedicoAdscrito (MedicoAdscritoJoinDTO medico_adscrito);

	/**
	   * Actualiza un médico adscrito en el sistema
	   * @param medico_adscrito Datos del medico adscrito
	   */
	public void actualizarMedicoAdscrito(MedicoAdscritoJoinDTO medico_adscrito);

	/**
	   * Lee los datos asociados a un médico adscrito
	   * @param id Identificador del médico adscrito
	   * @return El DTO con los datos del médico adscrito
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
	public MedicoAdscritoJoinDTO leerMedicoAdscrito(long id);
```

```
	/**
	   * Lee una lista de médicos adscritos
	   * @return La lista de DTOs con la información de los médicos adscritos
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
	public List<MedicoAdscritoJoinDTO> leerMedicosAdscritos();

	/**
	 * Lee una lista de medicos adscritos de acuerdo a un filtro de busqueda
	 * @param filter Filtro que fungira como patron de busqueda
	 * @return La lista de medicos adscritos que cumplieron con el filtro de
busqueda
	 */
	@Transactional( propagation=Propagation.SUPPORTS, readOnly = true )
	public List<MedicoAdscritoJoinDTO>
leerMedicosAdscritosFiltro( FiltroMedicosAdscritos filter );
}
```

8.1.6 AdministradorMedicosResidentes

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
package mx.uam.azc.pt.business;

import java.util.List;

import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import mx.uam.azc.pt.business.local.FiltroMedicosResidentes;
import mx.uam.azc.pt.data.ExamenJoinDTO;
import mx.uam.azc.pt.data.MedicoResidenteJoinDTO;
import mx.uam.azc.pt.data.MedicoResidenteSaveDTO;

/**
 * Interfase que provee de los métodos para la administración de Médicos
Residentes
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Transactional( rollbackFor=Exception.class )
public interface AdministradorMedicosResidentes {

	/**
	* Inserta un médico residente al sistema
	* @param administrador Datos del administrador
	*/
```

```
      public void insertarMedicoResidente (MedicoResidenteJoinDTO
medico_residente);

      /**
      * Inserta un médico residente al sistema
      * @param administrador Datos del administrador
      */
      public void insertarMedicoResidente (MedicoResidenteSaveDTO
medico_residente_save);

      /**
      * Inserta un médico residente al sistema con todo y exámenes
      * @param médico_residente Datos del médico residente con exámenes incluidos
      */
      public void insertarExamenes(MedicoResidenteJoinDTO medico_residente);

      /**
      * Asigna un examen a un médico residente
      * @param medico_residente Datos del médico residente
      * @param examen Datos del examen
      */
      public void asignarExamenes(MedicoResidenteJoinDTO medico_residente,
ExamenJoinDTO examen);

      /**
         * Actualiza un médico residente en el sistema
         * @param medico_residente Datos del médico residente
         */
      public void actualizarMedicoResidente(MedicoResidenteJoinDTO
medico_residente);

      /**
         * Actualiza un examen en el sistema
         * @param examen Datos del examen
         */
      public void actualizarExamen(ExamenJoinDTO examen);

      /**
         * Lee los datos asociados a un examen
         * @param id Identificador del examen
         * @return El DTO con los datos del examen
         */
      @Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
        public ExamenJoinDTO leerExamenId(long id);

      /**
         * Deshabilita los datos asociados a un médico residente
         * @param id Identificador del médico residente
         * @return El DTO con los datos del médico residente
         */
      @Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
      public void borrarMedicoResidente(long id);

      /**
```

```
         * Lee los datos asociados a un médicos residente
         * @param id Identificador del médico residente
         * @return El DTO con los datos del médico residente
         */
      @Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
        public MedicoResidenteJoinDTO leerMedicoResidenteId(long id);

      /**
         * Lee una lista de médicos residentes
         * @return La lista de DTOs con la información de los médicos residentes
         */
      @Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
      public List<MedicoResidenteJoinDTO> leerMedicosResidentes();

      /**
       * Metodo de prueba
       */
      //@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
      public void leerMedicosResidentes2();

      /**
       * Metodo que lee uan lista de medicos residentes de acuerdo a un filtro de
busqueda
       * @param filter Filtro que fungira como patron de busqueda
       * @return La lista de medicos residentes que cumplieron el patron de
busqueda
       */
      @Transactional( propagation=Propagation.SUPPORTS, readOnly = true )
      public List<MedicoResidenteJoinDTO>
leerMedicosResidentesFiltro( FiltroMedicosResidentes filter );

}
```

8.1.7 AdministradorPacientes

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
package mx.uam.azc.pt.business;

import java.util.List;

import mx.uam.azc.pt.business.local.FiltroAdministradores;
import mx.uam.azc.pt.business.local.FiltroPacientes;
import mx.uam.azc.pt.data.AdministradorJoinDTO;
import mx.uam.azc.pt.data.PacienteJoinDTO;

import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;
```

```java
/**
 * Interfase que provee de los métodos para la administración de Pacientes
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Transactional(rollbackFor=Exception.class)
public interface AdministradorPacientes {

	/**
	* Inserta un paciente al sistema
	* @param paciente Datos del paciente
	*/
	public void insertarPaciente(PacienteJoinDTO paciente);

	/**
	   * Actualiza un paciente en el sistema
	   * @param paciente Datos del paciente
	   */
	public void actualizacionPaciente(PacienteJoinDTO paciente);

	/**
	   * Lee los datos asociados a un paciente
	   * @param id Identificador del paciente
	   * @return El DTO con los datos del paciente
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
	public PacienteJoinDTO leerPaciente(long id);

	/**
	   * Lee una lista de pacientes
	   * @return La lista de DTOs con la información de los pacientes
	   */
	@Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
	public List<PacienteJoinDTO> leerPacientes();

	/**
	 * Metodo que lee una lista de pacientes de acuerdo a un filtro de busqueda
	 * @param filter Filtro que fungira como patron de busqueda
	 * @return La lista de pacientes que cumplieron el patron de busqueda
	 */
	@Transactional( propagation=Propagation.SUPPORTS, readOnly = true )
	public List<PacienteJoinDTO> leerPacientesFiltro( FiltroPacientes filter );

}
```

8.1.8 AdministradorUsuarios

```java
/**
 *  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
 *  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
```

```java
package mx.uam.azc.pt.business;

import java.util.List;
import java.util.Map;

import mx.uam.azc.pt.data.UsuarioDTO;

import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

/**
 * Interfase que provee de los métodos para la administración de Usuarios
 *
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
public interface AdministradorUsuarios {

	/**
	 * Inserta un usuario al sistema
	 *
	 * @param administrador
	 *            Datos del usuario
	 */
	public void insertarUsuario(UsuarioDTO usuario);

	/**
	 * Actualiza un usuario en el sistema
	 *
	 * @param usuario
	 *            Datos del usuario
	 */
	public void actualizarUsuario(UsuarioDTO usuario);

	/**
	 * Lee los datos asociados a usuario
	 *
	 * @param id
	 *            Identificador del usuario
	 * @return El DTO con los datos del usuario
	 */
	@Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
	public UsuarioDTO leerUsuario(long id);

	/**
	 * Metodo que lee una lista de usuarios
	 * @return La lista de todos los usuariis
	 */
	@Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
	public List<UsuarioDTO> leerUsuarios();

	/**
	 * Metodo que carga a los usuarios a un mapa
	 * @return El mapa con la informacion de los usuarios
```

```
         */
        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public Map<Long, String> cargarUsuarios();

        /**
         * Metodo para la lectura de login/password en la pantalla de login
         * @param login Login del usuario
         * @param password Passord del usuario
         * @return El ojeto UsuarioDTO de acuerdo a los valores proporcionados
         */
        @Transactional( propagation=Propagation.SUPPORTS, readOnly = true )
        public UsuarioDTO leerUsuarioLoginPassword(String login, String password);

}
```

8.2 mx.uam.azc.pt.business.local

8.2.1 AdministradorAdministradoresImpl

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
package mx.uam.azc.pt.business.local;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import mx.uam.azc.pt.business.AdministradorAdministradores;
import mx.uam.azc.pt.data.AdministradorJoinDTO;
import mx.uam.azc.pt.data.MedicoAdscritoJoinDTO;

/**
 * Clase que provee la implementación de los métodos de la interfase
AdministradorAdministradores
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
public class AdministradorAdministradoresImpl implements
            AdministradorAdministradores {

      /**
         * Inserta un administrador en el sistema
         * @param administrador Datos del administrador
         */
      public void insertarAdministrador(AdministradorJoinDTO administrador) {
```

```
            getSession().save(administrador);
      }

      /**
         * Actualiza un administrador en el sistema
         * @param administrador Datos del administrador
         */
      public void actualizarAdministrador(AdministradorJoinDTO administrador) {

            getSession().update(administrador);
      }

      /**
         * Lee los datos asociados a un administrador
         * @param id Identificador del administrador
         * @return El DTO con los datos del administrador
         */
      public AdministradorJoinDTO leerAdministrador(long id) {

            AdministradorJoinDTO administrador = new AdministradorJoinDTO();
            administrador = (AdministradorJoinDTO)
getSession().load(AdministradorJoinDTO.class, id);
            return administrador;
      }

      /**
         * Lee una lista de administradores
         * @return La lista de DTOs con la información de los administradores
         */
      public List<AdministradorJoinDTO> leerAdministradores() {

            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_administradores");
            return query.list();
      }

      /**
       * Metodo que lee una lista de administradores de acuerdo a un filtro de
busqueda
       */
      public List<AdministradorJoinDTO>
leerAdministradoresFiltro(FiltroAdministradores filter) {
            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_administradores_filtro");
            query.setString("nombres", "%" + filter.getNombres() + "%");
            System.out.println(query.getQueryString());
            return query.list();
      }

      /**
         * Session factory de Hibernate
         */
        private SessionFactory _sessionFactory;
```

```
        ////////////////////////// METODOS DE
CONFIGURACION ////////////////////////////

        public void setSessionFactory( SessionFactory sessionFactory )
        {
          _sessionFactory = sessionFactory;
        }

        ////////////////////// METODOS DE LAS INTERFAZ ////////////////////
        /**
         * Método que obtiene el Session Factory de Hibernate
         */
        private Session getSession()
        {
          return _sessionFactory.getCurrentSession();
        }
}
```

8.2.2 AdministradorCasosImpl

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
package mx.uam.azc.pt.business.local;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import mx.uam.azc.pt.business.AdministradorCasos;
import mx.uam.azc.pt.data.CasoJoinDTO;
import mx.uam.azc.pt.data.PacienteJoinDTO;

/**
 * Clase que provee la implementación de los métodos de la interfase
AdministradorCasos
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
public class AdministradorCasosImpl implements AdministradorCasos {

      /**
         * Inserta un caso en el sistema
         * @param caso Datos del caso
         */
      public void insertarCaso(CasoJoinDTO caso) {
```

```
            getSession().save(caso);
        }

        /**
           * Actualiza un caso en el sistema
           * @param caso Datos del caso
           */
        public void actualizarCaso(CasoJoinDTO caso) {

            getSession().update(caso);
        }

        /**
           * Lee una lista de casos
           * @return La lista de DTOs con la información de los casos
           */
        public List<CasoJoinDTO> leerCasos() {

            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_casos");
            return query.list();
        }

        /**
           * Lee los datos asociados a un caso
           * @param id Identificador del caso
           * @return El DTO con los datos del caso
           */
        public CasoJoinDTO leerCaso(long id) {

            CasoJoinDTO caso = new CasoJoinDTO();
            caso = (CasoJoinDTO)getSession().load(CasoJoinDTO.class, id);
            return caso;
        }

        /**
         * Metodo que lee una lista de casos de acuerdo a un filtro de busqueda
         */
        public List<CasoJoinDTO> leerCasosFiltro(FiltroCasos filter) {
            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_casos_filtro");
            query.setString("titulo", "%" + filter.getTitulo() + "%");
            System.out.println(query.getQueryString());
            return query.list();
        }

        /**
           * Session factory de Hibernate
           */
          private SessionFactory _sessionFactory;

          ////////////////////////// METODOS DE
CONFIGURACION ////////////////////////////
```

```
        public void setSessionFactory( SessionFactory sessionFactory )
        {
          _sessionFactory = sessionFactory;
        }

        ////////////////////// METODOS DE LAS INTERFAZ ////////////////////
        /**
         * Método que obtiene el Session Factory de Hibernate
         */
        private Session getSession()
        {
          return _sessionFactory.getCurrentSession();
        }
}
```

8.2.3 AdministradorContenidosImpl

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
package mx.uam.azc.pt.business.local;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import mx.uam.azc.pt.business.AdministradorContenidos;
import mx.uam.azc.pt.data.CasoJoinDTO;
import mx.uam.azc.pt.data.ContenidoJoinDTO;
import mx.uam.azc.pt.data.ReferenciaDTO;
import mx.uam.azc.pt.data.SubContenidoDTO;
import mx.uam.azc.pt.data.TemaRelacionadoJoinDTO;

/**
 * Clase que provee la implementación de los métodos de la interfase
AdministradorContenidos
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
public class AdministradorContenidosImpl implements AdministradorContenidos {

      /**
         * Inserta un contenido en el sistema
         * @param contenido Datos del contenido
         */
      public void insertarContenido(ContenidoJoinDTO contenido) {

            getSession().save(contenido);
```

```
	}

	/**
	   * Inserta un subcontenido en el sistema
	   * @param subcontenido Datos del subcontenido
	   */
	public void insertarSubContenido(SubContenidoDTO subcontenido) {

		getSession().save(subcontenido);
	}

	/**
	   * Inserta una referencia en el sistema
	   * @param referencia Datos de la referencia
	   */
	public void insertarReferencia(ReferenciaDTO referencia) {

		getSession().save(referencia);
	}

	/**
	   * Inserta un tema relacionado en el sistema
	   * @param tema_relacionado Datos del tema relacionado
	   */
	public void insertarTemaRelacionado(TemaRelacionadoJoinDTO tema_relacionado)
{

		getSession().save(tema_relacionado);
	}

	/**
	   * Actualiza un contenido en el sistema
	   * @param contenido Datos del contenido
	   */
	public void actualizarContenido(ContenidoJoinDTO contenido) {

		getSession().update(contenido);
	}

	/**
	   * Actualiza un subcontenido en el sistema
	   * @param subcontenido Datos del subcontenido
	   */
	public void actualizarSubContenido(SubContenidoDTO subcontenido) {

		getSession().update(subcontenido);
	}

	/**
	   * Actualiza una referencia en el sistema
	   * @param referencia Datos de la referencia
	   */
	public void actualizarReferencia(ReferenciaDTO referencia) {
```

```
            getSession().update(referencia);
      }

      /**
         * Actualiza un tema relacionado en el sistema
         * @param tema_relacionado Datos del tema relacionado
         */
      public void actualizarTemaRelacionado(TemaRelacionadoJoinDTO
tema_relacionado) {

            getSession().update(tema_relacionado);
      }

      /**
         * Lee los datos asociados a un contenido
         * @param id Identificador del contenido
         * @return El DTO con los datos del contenido
         */
      public ContenidoJoinDTO leerContenido(long id) {

            ContenidoJoinDTO contenido = new ContenidoJoinDTO();
            contenido=(ContenidoJoinDTO)getSession().load(ContenidoJoinDTO.class,
id);
            return contenido;
      }

      /**
         * Lee los datos asociados a un subcontenido
         * @param id Identificador del subcontenido
         * @return El DTO con los datos del subcontenido
         */
      public SubContenidoDTO leerSubContenido(long id) {

            SubContenidoDTO subcontenido = new SubContenidoDTO();
            subcontenido =
(SubContenidoDTO)getSession().load(SubContenidoDTO.class, id);
            return subcontenido;
      }

      /**
         * Lee los datos asociados a una referencia
         * @param id Identificador de la referencia
         * @return El DTO con los datos de la referencia
         */
      public ReferenciaDTO leerReferencia(long id) {

            ReferenciaDTO referencia = new ReferenciaDTO();
            referencia = (ReferenciaDTO)getSession().load(ReferenciaDTO.class, id);
            return referencia;
      }

      /**
         * Lee los datos asociados a un tema relacionado
         * @param id Identificador del tema relacionado
```

```
         * @return El DTO con los datos del tema relacionado
         */
        public TemaRelacionadoJoinDTO leerTemaRelacionado(long id) {

              TemaRelacionadoJoinDTO tema_relacionado = new TemaRelacionadoJoinDTO();
              tema_relacionado =
(TemaRelacionadoJoinDTO)getSession().load(TemaRelacionadoJoinDTO.class, id);
              return tema_relacionado;
        }

        /**
         * Lee una lista de contenidos
         * @return La lista de DTOs con la información de los contenidos
         */
        public List<ContenidoJoinDTO> leerContenidos() {
              Session session = _sessionFactory.getCurrentSession();
              Query query = session.getNamedQuery("leer_contenidos");
              return query.list();
        }

        /**
         * Lee una lista de subcontenidos
         * @return La lista de DTOs con la información de los subcontenidos
         */
        public List<SubContenidoDTO> leerSubContenidos() {

              Session session = _sessionFactory.getCurrentSession();
              Query query = session.getNamedQuery("leer_subcontenidos");
              return query.list();
        }

        /**
         * Lee una lista de referencia
         * @return La lista de DTOs con la información de las referencias
         */
        public List<ReferenciaDTO> leerReferencias() {

              Session session = _sessionFactory.getCurrentSession();
              Query query = session.getNamedQuery("leer_referencias");
              return query.list();
        }

        /**
         * Lee una lista de temas relacionados
         * @return La lista de DTOs con la información de los temas relacionados
         */
        public List<TemaRelacionadoJoinDTO> leerTemasRelacionados() {

              Session session = _sessionFactory.getCurrentSession();
              Query query = session.getNamedQuery("leer_temas_relacionados");
              return query.list();
        }

        /**
```

```
       * Metodo que lee una lista de contenidos de acuerdo a un filtro de busqueda
       */
      public List<ContenidoJoinDTO> leerContenidosFiltro(FiltroContenidos filter)
{
            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_contenidos_filtro");
            query.setString("titulo_contenido", "%" + filter.getTitulo_contenido()
+ "%");
            System.out.println(query.getQueryString());
            return query.list();
      }

      /**
         * Session factory de Hibernate
         */
        private SessionFactory _sessionFactory;

        ////////////////////////// METODOS DE
CONFIGURACION ////////////////////////////

        public void setSessionFactory( SessionFactory sessionFactory )
        {
          _sessionFactory = sessionFactory;
        }

        ////////////////////// METODOS DE LAS INTERFAZ ////////////////////
        /**
         * Método que obtiene el Session Factory de Hibernate
         */
        private Session getSession()
        {
          return _sessionFactory.getCurrentSession();
        }
}
```

8.2.4 AdministradorExamenesImpl

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
package mx.uam.azc.pt.business.local;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import mx.uam.azc.pt.business.AdministradorExamenes;
import mx.uam.azc.pt.data.ContenidoJoinDTO;
```

```java
import mx.uam.azc.pt.data.ExamenJoinDTO;
import mx.uam.azc.pt.data.MedicoAdscritoJoinDTO;
import mx.uam.azc.pt.data.PreguntaDTO;
import mx.uam.azc.pt.data.RespuestaDTO;

/**
 * Clase que provee la implementación de los métodos de la interfase
AdministradorExamenes
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
public class AdministradorExamenesImpl implements AdministradorExamenes {

	/**
	   * Inserta un examen en el sistema
	   * @param examen Datos del examen
	   */
	public void insertarExamen(ExamenJoinDTO examen) {

		getSession().save(examen);
	}

	/**
	   * Inserta una pregunta en el sistema
	   * @param pregunta Datos de la pregunta
	   */
	public void insertarPregunta(PreguntaDTO pregunta) {

		getSession().save(pregunta);
	}

	/**
	   * Inserta una respuesta en el sistema
	   * @param respuesta Datos de la respuesta
	   */
	public void insertarRespuesta(RespuestaDTO respuesta) {

		getSession().save(respuesta);
	}

	/**
	   * Actualiza un examen en el sistema
	   * @param examen Datos del examen
	   */
	public void actualizarExamen(ExamenJoinDTO examen) {

		getSession().update(examen);
	}

	/**
	   * Actualiza una pregunta en el sistema
	   * @param pregunta Datos de la pregunta
	   */
	public void actualizarPregunta(PreguntaDTO pregunta) {
```

```
        getSession().update(pregunta);
    }

    /**
     * Actualiza una respuesta en el sistema
     * @param respuesta Datos de la respuesta
     */
    public void actualizarRespuesta(RespuestaDTO respuesta) {

        getSession().update(respuesta);
    }

    /**
     * Lee los datos asociados a un examen
     * @param id Identificador del examen
     * @return El DTO con los datos del examen
     */
    public ExamenJoinDTO leerExamen(long id) {

        ExamenJoinDTO examen = new ExamenJoinDTO();
        examen = (ExamenJoinDTO) getSession().load(ExamenJoinDTO.class, id);
        return examen;
    }

    /**
     * Lee los datos asociados a una pregunta
     * @param id Identificador de la pregunta
     * @return El DTO con los datos de la pregunta
     */
    public PreguntaDTO leerPregunta(long id) {

        PreguntaDTO pregunta = new PreguntaDTO();
        pregunta = (PreguntaDTO)getSession().load(PreguntaDTO.class, id);
        return pregunta;
    }

    /**
     * Lee los datos asociados a una respuesta
     * @param id Identificador de la respuesta
     * @return El DTO con los datos de la respuesta
     */
    public RespuestaDTO leerRespuesta(long id) {

        RespuestaDTO respuesta = new RespuestaDTO();
        respuesta = (RespuestaDTO)getSession().load(RespuestaDTO.class, id);
        return respuesta;
    }

    /**
     * Lee una lista de exámenes
     * @return La lista de DTOs con la información de los exámenes
     */
    public List<ExamenJoinDTO> leerExamenes() {
```

```
            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_examenes");
            return query.list();
      }

      /**
         * Lee una lista de preguntas
         * @return La lista de DTOs con la información de las preguntas
         */
      public List<PreguntaDTO> leerPreguntas() {

            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_preguntas");
            return query.list();
      }

      /**
         * Lee una lista de respuestas
         * @return La lista de DTOs con la información de las respuestas
         */
      public List<RespuestaDTO> leerRespuestas() {

            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_respuestas");
            return query.list();
      }

      /**
       * Metodo que lee una lista de examenes de acuerdo a un filtro de busqueda
       */
      public List<ExamenJoinDTO> leerExamenesFiltro(FiltroExamenes filter) {
            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_examenes_filtro");
            query.setString("titulo", "%" + filter.getTitulo() + "%");
            System.out.println(query.getQueryString());
            return query.list();
      }

      /**
         * Session factory de Hibernate
         */
        private SessionFactory _sessionFactory;

        ////////////////////////// METODOS DE
CONFIGURACION ////////////////////////////

        public void setSessionFactory( SessionFactory sessionFactory )
        {
          _sessionFactory = sessionFactory;
        }

        ////////////////////// METODOS DE LAS INTERFAZ ////////////////////
        /**
```

```
         * Método que obtiene el Session Factory de Hibernate
         */
        private Session getSession()
        {
          return _sessionFactory.getCurrentSession();
        }

}
```

8.2.5 AdministradorMedicosAdscritosImpl

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
package mx.uam.azc.pt.business.local;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import mx.uam.azc.pt.business.AdministradorMedicosAdscritos;
import mx.uam.azc.pt.data.ExamenJoinDTO;
import mx.uam.azc.pt.data.MedicoAdscritoJoinDTO;
import mx.uam.azc.pt.data.MedicoResidenteJoinDTO;

/**
 * Clase que provee la implementación de los métodos de la interfase
AdministradorMedicosAdscritos
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
public class AdministradorMedicosAdscritosImpl implements
            AdministradorMedicosAdscritos {

      /**
       * @param medico_adscrito
       * Metodo que inserta a un médico adscrito a la Base de Datos
       */
      public void insertarMedicoAdscrito(MedicoAdscritoJoinDTO medico_adscrito) {
            getSession().save(medico_adscrito);
      }

      /**
         * Actualiza un médico adscrito en el sistema
         * @param medico_adscrito Datos del medico_adscrito
         */
      public void actualizarMedicoAdscrito(MedicoAdscritoJoinDTO medico_adscrito)
{
```

```
            getSession().update(medico_adscrito);
      }

      /**
         * Lee los datos asociados a un médico adscrito
         * @param id Identificador del médico adscrito
         * @return El DTO con los datos del médico adscrito
         */
      public MedicoAdscritoJoinDTO leerMedicoAdscrito(long id) {

            MedicoAdscritoJoinDTO medico_adscrito = new MedicoAdscritoJoinDTO();
            medico_adscrito = (MedicoAdscritoJoinDTO)
getSession().load(MedicoAdscritoJoinDTO.class, id);
            return medico_adscrito;
      }

      /**
         * Lee una lista de médicos adscritos
         * @return La lista de DTOs con la información de los médicos adscritos
         */
      public List<MedicoAdscritoJoinDTO> leerMedicosAdscritos() {
            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_medicos_adscritos");
            return query.list();
      }

      /**
       * Metodo que lee una lista de metodos adscritos de acuerdo a un filtro de
busqueda
       */
      public List<MedicoAdscritoJoinDTO>
leerMedicosAdscritosFiltro(FiltroMedicosAdscritos filter) {
            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_medicos_adscritos_filtro");
            query.setString("nombres", "%" + filter.getNombres() + "%");
            System.out.println(query.getQueryString());
            return query.list();
      }

      /**
         * Session factory de Hibernate
         */
        private SessionFactory _sessionFactory;

        ////////////////////////// METODOS DE
CONFIGURACION ////////////////////////////

        public void setSessionFactory( SessionFactory sessionFactory )
        {
          _sessionFactory = sessionFactory;
        }

        ////////////////////// METODOS DE LAS INTERFAZ ////////////////////
```

```
        /**
         * Método que obtiene el Session Factory de Hibernate
         */
        private Session getSession()
        {
          return _sessionFactory.getCurrentSession();
        }
}
```

8.2.6 AdministradorMedicosResidentesImpl

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
package mx.uam.azc.pt.business.local;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import mx.uam.azc.pt.business.AdministradorMedicosResidentes;
import mx.uam.azc.pt.data.ExamenJoinDTO;
import mx.uam.azc.pt.data.MedicoResidenteJoinDTO;
import mx.uam.azc.pt.data.MedicoResidenteSaveDTO;

/**
 * Clase que provee la implementación de los métodos de la interfase
AdministradorMedicosResidentes
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
public class AdministradorMedicosResidentesImpl implements
            AdministradorMedicosResidentes {

      /**
         * Inserta un médico residente en el sistema
         * @param medico_residente Datos del médico residente
         */
      public void insertarMedicoResidente(MedicoResidenteJoinDTO medico_residente)
{

            getSession().save(medico_residente);
      }

      /**
         * Inserta un médico residente en el sistema
         * @param medico_residente Datos del médico residente
         */
```

```
	public void insertarMedicoResidente(MedicoResidenteSaveDTO
medico_residente_save) {

		getSession().save(medico_residente_save);
	}

	/**
	   * Inserta un médico residente  con todo y exámenes en el sistema
	   * @param medico_residente Datos del médico residente con todo y exámenes
	   */
	public void insertarExamenes(MedicoResidenteJoinDTO medico_residente) {
		getSession().saveOrUpdate(medico_residente);
	}

	/**
	   * Actualiza un examen en el sistema
	   * @param examen Datos del examen
	   */
	public void actualizarExamen(ExamenJoinDTO examen) {

		getSession().update(examen);
	}

	/**
	   * Actualiza un médico residente en el sistema
	   * @param medico_residente Datos del médico residente
	   */
	public void actualizarMedicoResidente(MedicoResidenteJoinDTO
medico_residente) {

		getSession().update(medico_residente);
	}

	/**
	   * Asigna un examen a un médico residente
	   * @param medico_residente Datos del médico residente
	   * @param examen Datos del examen
	   */
	public void asignarExamenes(MedicoResidenteJoinDTO medico_residente,
			ExamenJoinDTO examen) {

		medico_residente.agregarExamen(examen);
		getSession().update(medico_residente);
	}

	/**
	   * Lee los datos asociados a un médico residente
	   * @param id Identificador del médico residente
	   * @return El DTO con los datos del médico residente
	   */
	public MedicoResidenteJoinDTO leerMedicoResidenteId(long id) {

		System.out.println("leer_medico_residente");
		MedicoResidenteJoinDTO medico_residente = new MedicoResidenteJoinDTO();
```

```
            medico_residente = (MedicoResidenteJoinDTO)
getSession().load(MedicoResidenteJoinDTO.class, id);
            return medico_residente;
      }

      /**
         * Lee los datos asociados a un examen
         * @param id Identificador del examen
         * @return El DTO con los datos del examen
         */
      public ExamenJoinDTO leerExamenId(long id) {
            System.out.println("leido1");
            ExamenJoinDTO examen = new ExamenJoinDTO();
            examen = (ExamenJoinDTO)getSession().load(ExamenJoinDTO.class, id);
            return examen;
      }

      /**
         * Deshabilita a un médico residente del sistema
         * @param id Identificador del médico residente
         */
      public void borrarMedicoResidente(long id) {
            MedicoResidenteJoinDTO medico_residente = new MedicoResidenteJoinDTO();
            medico_residente = (MedicoResidenteJoinDTO)
getSession().load(MedicoResidenteJoinDTO.class, id);
            medico_residente.setStatus(1);
      }

      /**
         * Lee una lista de médicos residentes
         * @return La lista de DTOs con la información de los médicos residentes
         */
      public List<MedicoResidenteJoinDTO> leerMedicosResidentes() {
            System.out.println("leidos_medicos_residentes");
            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_medicos_residentes");
            System.out.println("acabe el metodo de leer medicos residentes");
            return query.list();
      }

      /**
       * Metodo que lee una lista de medicos residentes de acuerdo a un filtro de
busqueda
       */
      public List<MedicoResidenteJoinDTO>
leerMedicosResidentesFiltro(FiltroMedicosResidentes filter) {
            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_medicos_residentes_filtro");
            query.setString("nombres", "%" + filter.getNombres() + "%");
            System.out.println(query.getQueryString());
            return query.list();
      }

      public void leerMedicosResidentes2() {
```

```
            System.out.println("empieza_lectura_medicos_residentes");
            System.out.println("acabe el metodo de leer medicos residentes");
      }

      /**
         * Session factory de Hibernate
         */
        private SessionFactory _sessionFactory;

        ////////////////////////// METODOS DE
CONFIGURACION ////////////////////////////

        public void setSessionFactory( SessionFactory sessionFactory )
        {
          _sessionFactory = sessionFactory;
        }

        ////////////////////// METODOS DE LAS INTERFAZ ////////////////////
        /**
         * Método que obtiene el Session Factory de Hibernate
         */
        private Session getSession()
        {
          return _sessionFactory.getCurrentSession();
        }
}
```

8.2.7 AdministradorPacientesImpl

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
package mx.uam.azc.pt.business.local;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import mx.uam.azc.pt.business.AdministradorPacientes;
import mx.uam.azc.pt.data.AdministradorJoinDTO;
import mx.uam.azc.pt.data.PacienteJoinDTO;

/**
 * Clase que provee la implementación de los métodos de la interfase
AdministradorPaciente
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
```

```
public class AdministradorPacientesImpl implements AdministradorPacientes {

	/**
	 * Inserta un paciente en el sistema
	 * @param paciente Datos del paciente
	 */
	public void insertarPaciente(PacienteJoinDTO paciente) {

		getSession().save(paciente);
	}

	/**
	 * Actualiza un paciente en el sistema
	 * @param paciente Datos del paciente
	 */
	public void actualizacionPaciente(PacienteJoinDTO paciente) {

		getSession().update(paciente);
	}

	/**
	 * Lee los datos asociados a un paciente
	 * @param id Identificador del paciente
	 * @return El DTO con los datos del paciente
	 */
	public PacienteJoinDTO leerPaciente(long id) {

		PacienteJoinDTO paciente =  new PacienteJoinDTO();
		paciente = (PacienteJoinDTO)getSession().load( PacienteJoinDTO.class,
id );
		return paciente;
	}

	/**
	 * Lee una lista de paciente
	 * @return La lista de DTOs con la información de los pacientes
	 */
	public List<PacienteJoinDTO> leerPacientes() {
		Session session = _sessionFactory.getCurrentSession();
		Query query = session.getNamedQuery("leer_pacientes");
		return query.list();
	}

	/**
	 * Metodo que lee una lista de pacientes de acuerdo a un patron de busqueda
	 */
	public List<PacienteJoinDTO> leerPacientesFiltro(FiltroPacientes filter) {
		Session session = _sessionFactory.getCurrentSession();
		Query query = session.getNamedQuery("leer_pacientes_filtro");
		query.setString("nombres", "%" + filter.getNombres() + "%");
		System.out.println(query.getQueryString());
		return query.list();
	}
```

```
      /**
         * Session factory de Hibernate
         */
        private SessionFactory _sessionFactory;

        ////////////////////////// METODOS DE
CONFIGURACION ////////////////////////////

        public void setSessionFactory( SessionFactory sessionFactory )
        {
          _sessionFactory = sessionFactory;
        }

        ////////////////////// METODOS DE LAS INTERFAZ ////////////////////
        /**
         * Método que obtiene el Session Factory de Hibernate
         */
        private Session getSession()
        {
          return _sessionFactory.getCurrentSession();
        }

}
```

8.2.8 AdministradorUsuariosImpl

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */
package mx.uam.azc.pt.business.local;

import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

import mx.uam.azc.pt.business.AdministradorUsuarios;
import mx.uam.azc.pt.data.UsuarioDTO;

/**
 * Clase que provee la implementación de los métodos de la interfase
AdministradorUsuarios
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
public class AdministradorUsuariosImpl implements AdministradorUsuarios {
```

```java
/**
   * Inserta un usuario en el sistema
   * @param usuario Datos del usuario
   */
public void insertarUsuario(UsuarioDTO usuario) {

      getSession().save(usuario);
}

/**
   * Lee los datos asociados a un usuario
   * @param id Identificador del usuario
   * @return El DTO con los datos del usuario
   */
public UsuarioDTO leerUsuario(long id) {

      UsuarioDTO usuario = new UsuarioDTO();
      usuario = (UsuarioDTO)getSession().load(UsuarioDTO.class, id);
      return usuario;
}

/**
   * Actualiza un usuario en el sistema
   * @param usuario Datos del usuario
   */
public void actualizarUsuario(UsuarioDTO usuario) {

      getSession().update(usuario);

}

/**
 * Metodo que lee todos los usuarios del sistema y los regresa en una lista
 */
public List<UsuarioDTO> leerUsuarios() {
      Session session = _sessionFactory.getCurrentSession();
      Query query = session.getNamedQuery( "leer_usuarios" );
      return query.list();

}

/**
 * Metodo que carga a los usuarios en un mapa long string
 */
public Map<Long, String> cargarUsuarios() {

      List<UsuarioDTO> t = leerUsuarios();
      Iterator<UsuarioDTO> i = t.iterator();
      UsuarioDTO tt = new UsuarioDTO();
      Map<Long,String> mapa = new TreeMap<Long,String>();

      while(i.hasNext()){
            tt = i.next();
```

```
                mapa.put( new Long(tt.getId()),tt.getNombres() );
            }
            return mapa;
      }

      /**
       * Metodo que lee un usuario dependiendo de la autenticacion de cada usuario
       */
      public UsuarioDTO leerUsuarioLoginPassword(String login, String password) {
            Session session = _sessionFactory.getCurrentSession();
            Query query = session.getNamedQuery("leer_usuario_login_pass");
            query.setString(0, login);
            query.setString(1, password);
            List<UsuarioDTO> lista = query.list();
            return lista.isEmpty() ? null : lista.get(0);
      }

      /**
         * Session factory de Hibernate
         */
        private SessionFactory _sessionFactory;

        ////////////////////////// METODOS DE
CONFIGURACION ////////////////////////////

        public void setSessionFactory( SessionFactory sessionFactory )
        {
          _sessionFactory = sessionFactory;
        }

        ////////////////////////// METODOS DE LAS INTERFAZ ////////////////////
        /**
         * Método que obtiene el Session Factory de Hibernate
         */
        private Session getSession()
        {
          return _sessionFactory.getCurrentSession();
        }
}
```

8.2.9 FiltroAdministradores

```
package mx.uam.azc.pt.business.local;

/**
 * Clase que actuara como filtro de busqueda para administradores
 * @author Roberto Gonzalez
 *
 */
public class FiltroAdministradores {
```

```
	/**
	 * Cadena que sera el patron de busqueda para el filtro de administradores
	 */
	private String nombres;

	/**
	 * @return the nombres
	 */
	public String getNombres() {
		return nombres;
	}

	/**
	 * @param nombres the nombres to set
	 */
	public void setNombres(String nombres) {
		this.nombres = nombres;
	}

}
```

8.2.10 FiltroCasos

```
package mx.uam.azc.pt.business.local;

/**
 * Clase que actuara como filtro de busqueda para casos
 * @author Roberto Gonzalez
 *
 */
public class FiltroCasos {

	/**
	 * Cadena que sera el patron de busqueda para el filtro de casos
	 */
	private String titulo;

	/**
	 * @return the titulo
	 */
	public String getTitulo() {
		return titulo;
	}

	/**
	 * @param titulo the titulo to set
	 */
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
```

```
}
```

8.2.11 FiltroContenidos

```
package mx.uam.azc.pt.business.local;

/**
 * Clase que actuara como filtro de busqueda para contenidos
 * @author Roberto Gonzalez
 *
 */
public class FiltroContenidos {

	/**
	 * Cadena que sera el patron de busqueda para el filtro de contenidos
	 */
	private String titulo_contenido;

	/**
	 * @return the titulo_contenido
	 */
	public String getTitulo_contenido() {
		return titulo_contenido;
	}

	/**
	 * @param titulo the titulo_contenido to set
	 */
	public void setTitulo_contenido(String titulo_contenido) {
		this.titulo_contenido = titulo_contenido;
	}


}
```

8.2.12 FiltroExamenes

```
package mx.uam.azc.pt.business.local;

/**
 * Clase que actuara como filtro de busqueda para examenes
 * @author Roberto Gonzalez
 *
 */
public class FiltroExamenes {
```

```
	/**
	 * Cadena que sera el patron de busqueda para el filtro de examenes
	 */
	private String titulo;

	/**
	 * @return the titulo
	 */
	public String getTitulo() {
		return titulo;
	}

	/**
	 * @param titulo the titulo to set
	 */
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}


}
```

8.2.13 FiltroMedicosAdscritos

```
package mx.uam.azc.pt.business.local;

/**
 * Clase que actuara como filtro de busqueda medicos adscritos
 * @author Roberto Gonzalez
 *
 */
public class FiltroMedicosAdscritos {

	/**
	 * Cadena que sera el patron de busqueda para el filtro de medicos adscritos
	 */
	private String nombres;

	/**
	 * @return the nombres
	 */
	public String getNombres() {
		return nombres;
	}

	/**
	 * @param nombres the nombres to set
	 */
	public void setNombres(String nombres) {
```

```
            this.nombres = nombres;
      }

}
```

8.2.14 FiltroMedicosResidentes

```
package mx.uam.azc.pt.business.local;

/**
 * Clase que actuara como filtro de busqueda para medicos residentes
 * @author Roberto Gonzalez
 *
 */
public class FiltroMedicosResidentes {

      /**
       * Cadena que sera el patron de busqueda para el filtro de medicos residentes
       */
      private String nombres;

      /**
       * @return the nombres
       */
      public String getNombres() {
            return nombres;
      }

      /**
       * @param nombres the nombres to set
       */
      public void setNombres(String nombres) {
            this.nombres = nombres;
      }

}
```

8.2.15 FiltroPacientes

```
package mx.uam.azc.pt.business.local;

/**
 * Clase que actuara como filtro de busqueda para pacientes
 * @author Roberto Gonzalez
 *
 */
public class FiltroPacientes {

      /**
```

```
	 * Cadena que sera el patron de busqueda para el filtro de pacientes
	 */
	private String nombres;

	/**
	 * @return the nombres
	 */
	public String getNombres() {
		return nombres;
	}

	/**
	 * @param nombres the nombres to set
	 */
	public void setNombres(String nombres) {
		this.nombres = nombres;
	}

}
```

8.3 mx.uam.azc.pt.data

8.3.1 AdministradorJoinDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/

package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la información de un administrador en el sistema, hereda de
la clase UsuarioDTO
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name = "administrador")
```

```
@PrimaryKeyJoinColumn(name="id")
@Proxy( lazy=false )
public class AdministradorJoinDTO extends UsuarioDTO implements Serializable {

	/**
	 * email alternativo del medico residente, mapeado con el campo
-email_aternativo- de la tabla -medicos_residentes- de la Base de Datos
	 */
	private String email_alternativo;

	/**
	 * @return the email_alternativo
	 */
	@Column(name="email_alternativo")
	public String getEmail_alternativo() {
		return email_alternativo;
	}

	/**
	 * @param email_alternativo the email_alternativo to set
	 */
	public void setEmail_alternativo(String email_alternativo) {
		this.email_alternativo = email_alternativo;
	}
}
```

8.3.2 AdministradorSaveDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/

package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la información de un administrador en el sistema, hereda de
la clase UsuarioDTO
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
```

```
@Entity
@Table(name = "administrador")
@PrimaryKeyJoinColumn(name="id")
@Proxy( lazy=false )
public class AdministradorSaveDTO extends UsuarioDTO implements Serializable {

	/**
	 * email alternativo del medico residente, mapeado con el campo
-email_aternativo- de la tabla -medicos_residentes- de la Base de Datos
	 */
	private String email_alternativo;

	/**
	 * @return the email_alternativo
	 */
	@Column(name="email_alternativo")
	public String getEmail_alternativo() {
		return email_alternativo;
	}

	/**
	 * @param email_alternativo the email_alternativo to set
	 */
	public void setEmail_alternativo(String email_alternativo) {
		this.email_alternativo = email_alternativo;
	}
}
```

8.3.3 CasoJoinDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */

package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la información de un Caso en el sistema.
```

```
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name="casos")
@Proxy (lazy=false)
public class CasoJoinDTO implements Serializable {

	/**
	 * id_caso del caso, mapeado con el campo -id_casos- de la tabla -casos- de la Base de Datos
	 */
	private long id;
	/**
	 * titulo del caso, mapeado con el campo -titulos- de la tabla -casos- de la Base de Datos
	 */
	private String titulo;
	/**
	 * id_caso del contenido_caso, mapeado con el campo -contenido_caso- de la tabla -casos- de la Base de Datos
	 */
	private String contenido_caso;
	/**
	 * status del contenido_caso, mapeado con el campo -status- de la tabla -casos- de la Base de Datos
	 */
	private int status;

	////////////////////////////////////////////////////////////////////////////////

	/**
	 * @return the status
	 */
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @param status the status to set
	 */
	public void setStatus(int status) {
		this.status = status;
	}
	/**
	 * @return the id_caso
	 */
	@Id
	@Column(name="id")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId() {
		return id;
	}
```

```
	/**
	 * @return the titulo
	 */
	@Column(name="titulo")
	public String getTitulo() {
		return titulo;
	}
	/**
	 * @return the contenido_caso
	 */
	@Column(name="contenido_caso")
	public String getContenido_caso() {
		return contenido_caso;
	}
	/**
	 * @param id_caso the id_caso to set
	 */
	public void setId(long id) {
		this.id = id;
	}
	/**
	 * @param titulo the titulo to set
	 */
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
	/**
	 * @param contenido_caso the contenido_caso to set
	 */
	public void setContenido_caso(String contenido_caso) {
		this.contenido_caso = contenido_caso;
	}

}
```

8.3.4 CasoSaveDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
 */

package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

```java
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la información de un Caso en el sistema.
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name="casos")
@Proxy (lazy=false)
public class CasoSaveDTO implements Serializable {

	/**
	 * id_caso del caso, mapeado con el campo -id_casos- de la tabla -casos- de la Base de Datos
	 */
	private long id;
	/**
	 * titulo del caso, mapeado con el campo -titulos- de la tabla -casos- de la Base de Datos
	 */
	private String titulo;
	/**
	 * id_caso del contenido_caso, mapeado con el campo -contenido_caso- de la tabla -casos- de la Base de Datos
	 */
	private String contenido_caso;
	/**
	 * status del contenido_caso, mapeado con el campo -status- de la tabla -casos- de la Base de Datos
	 */
	private int status;

	//////////////////////////////////////////////////////////////////////////////

	/**
	 * @return the status
	 */
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @param status the status to set
	 */
	public void setStatus(int status) {
		this.status = status;
	}
	/**
	 * @return the id_caso
	 */
```

```
	@Id
	@Column(name="id")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId() {
		return id;
	}
	/**
	 * @return the titulo
	 */
	@Column(name="titulo")
	public String getTitulo() {
		return titulo;
	}
	/**
	 * @return the contenido_caso
	 */
	@Column(name="contenido_caso")
	public String getContenido_caso() {
		return contenido_caso;
	}
	/**
	 * @param id_caso the id_caso to set
	 */
	public void setId(long id) {
		this.id = id;
	}
	/**
	 * @param titulo the titulo to set
	 */
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
	/**
	 * @param contenido_caso the contenido_caso to set
	 */
	public void setContenido_caso(String contenido_caso) {
		this.contenido_caso = contenido_caso;
	}

}
```

8.3.5 ContenidoJoinDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;
import java.util.List;

import javax.persistence.CascadeType;
```

```java
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la informacion de un contenido en el sistema
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name="contenidos")
@Proxy(lazy=false)
public class ContenidoJoinDTO implements Serializable {

	/**
	 * id del contenido, mapeado con el campo -id_contenidos- de la tabla
-contenidos- de la Base de Datos
	 */
	private long id_contenidos;

	/**
	 * titulo del contenido, mapeado con el campo -titulo_contenido- de la tabla
-contenidos- de la Base de Datos
	 */
	private String titulo_contenido;

	/**
	 * descripcion del contenido, mapeado con el campo -descripcion- de la tabla
-contenidos- de la Base de Datos
	 */
	private String descripcion;

	/**
	 * subcontenidos asociados al contenido, hace referencia a la tabla
-subcontenidos- de la Base de Datos, mapeada por el campo -id_contenidos-
	 */
	private List<SubContenidoDTO> subcontenidos;

	/**
	 * referencias asociadas al contenido, hace referencia a la tabla
-referencias- de la Base de Datos, mapeada por el campo -id_contenidos-
	 */
	private List<ReferenciaDTO> referencias;

	/**
	 * temas relacionados asociados al contenido, hace referencia a la tabla
-temas_relacionados- de la Base de Datos, mapeada por el campo -id_contenidos-
```

```
	 */
	private List<TemaRelacionadoJoinDTO> temas_relacionados;
	/**
	 * status del contenido, mapeado con el campo -status- de la tabla
-contenidos- de la Base de Datos
	 */
	private int status;

	////////////////////////////////////////////////////////////////////////////
/////

	/**
	 * @return the id_contenidos
	 */
	@Id
	@Column(name="id_contenidos")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId_contenidos() {
		return id_contenidos;
	}
	/**
	 * @return the titulo_contenido
	 */
	@Column(name="titulo_contenido")
	public String getTitulo_contenido() {
		return titulo_contenido;
	}
	/**
	 * @return the descripcion
	 */
	@Column(name="descripcion")
	public String getDescripcion() {
		return descripcion;
	}
	/**
	 * @return the status
	 */
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @return the subcontenidos
	 */
	@OneToMany( cascade = CascadeType.ALL)
	@JoinColumn(name="id_contenidos")
	public List<SubContenidoDTO> getSubcontenidos() {
		return subcontenidos;
	}
	/**
	 * @param id_contenidos the id_contenidos to set
	 */
	public void setId_contenidos(long id_contenidos) {
		this.id_contenidos = id_contenidos;
```

```
	}
	/**
	 * @param titulo_contenido the titulo_contenido to set
	 */
	public void setTitulo_contenido(String titulo_contenido) {
		this.titulo_contenido = titulo_contenido;
	}
	/**
	 * @param descripcion the descripcion to set
	 */
	public void setDescripcion(String descripcion) {
		this.descripcion = descripcion;
	}
	/**
	 * @param status the status to set
	 */
	public void setStatus(int status) {
		this.status = status;
	}
	/**
	 * @param subcontenidos the subcontenidos to set
	 */
	public void setSubcontenidos(List<SubContenidoDTO> subcontenidos) {
		this.subcontenidos = subcontenidos;
	}
	/**
	 * @return the referencias
	 */
	@OneToMany( cascade = CascadeType.ALL)
	@JoinColumn(name="id_contenidos")
	public List<ReferenciaDTO> getReferencias() {
		return referencias;
	}
	/**
	 * @param referencias the referencias to set
	 */
	public void setReferencias(List<ReferenciaDTO> referencias) {
		this.referencias = referencias;
	}
	/**
	 * @return the temas_relacionados
	 */
	@OneToMany( cascade = CascadeType.ALL)
	@JoinColumn(name="id_contenidos")
	public List<TemaRelacionadoJoinDTO> getTemas_relacionados() {
		return temas_relacionados;
	}
	/**
	 * @param temas_relacionados the temas_relacionados to set
	 */
	public void setTemas_relacionados(List<TemaRelacionadoJoinDTO>
temas_relacionados) {
		this.temas_relacionados = temas_relacionados;
	}
```

```
}
```

8.3.6 ContenidoSaveDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la informacion de un contenido en el sistema
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name="contenidos")
@Proxy(lazy=false)
public class ContenidoSaveDTO implements Serializable {

	/**
	 * id del contenido, mapeado con el campo -id_contenidos- de la tabla
-contenidos- de la Base de Datos
	 */
	private long id_contenidos;

	/**
	 * titulo del contenido, mapeado con el campo -titulo_contenido- de la tabla
-contenidos- de la Base de Datos
	 */
	private String titulo_contenido;

	/**
	 * descripcion del contenido, mapeado con el campo -descripcion- de la tabla
-contenidos- de la Base de Datos
```

```
	 */
	private String descripcion;

	/**
	 * subcontenidos asociados al contenido, hace referencia a la tabla
-subcontenidos- de la Base de Datos, mapeada por el campo -id_contenidos-
	 */
	private List<SubContenidoDTO> subcontenidos;

	/**
	 * referencias asociadas al contenido, hace referencia a la tabla
-referencias- de la Base de Datos, mapeada por el campo -id_contenidos-
	 */
	private List<ReferenciaDTO> referencias;

	/**
	 * temas relacionados asociados al contenido, hace referencia a la tabla
-temas_relacionados- de la Base de Datos, mapeada por el campo -id_contenidos-
	 */
	private List<TemaRelacionadoJoinDTO> temas_relacionados;
	/**
	 * status del contenido, mapeado con el campo -status- de la tabla
-contenidos- de la Base de Datos
	 */
	private int status;

	////////////////////////////////////////////////////////////////////////
/////

	/**
	 * @return the id_contenidos
	 */
	@Id
	@Column(name="id_contenidos")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId_contenidos() {
		return id_contenidos;
	}
	/**
	 * @return the titulo_contenido
	 */
	@Column(name="titulo_contenido")
	public String getTitulo_contenido() {
		return titulo_contenido;
	}
	/**
	 * @return the descripcion
	 */
	@Column(name="descripcion")
	public String getDescripcion() {
		return descripcion;
	}
	/**
	 * @return the status
```

```
	 */
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @return the subcontenidos
	 */
	@OneToMany( cascade = CascadeType.ALL)
	@JoinColumn(name="id_contenidos")
	public List<SubContenidoDTO> getSubcontenidos() {
		return subcontenidos;
	}
	/**
	 * @param id_contenidos the id_contenidos to set
	 */
	public void setId_contenidos(long id_contenidos) {
		this.id_contenidos = id_contenidos;
	}
	/**
	 * @param titulo_contenido the titulo_contenido to set
	 */
	public void setTitulo_contenido(String titulo_contenido) {
		this.titulo_contenido = titulo_contenido;
	}
	/**
	 * @param descripcion the descripcion to set
	 */
	public void setDescripcion(String descripcion) {
		this.descripcion = descripcion;
	}
	/**
	 * @param status the status to set
	 */
	public void setStatus(int status) {
		this.status = status;
	}
	/**
	 * @param subcontenidos the subcontenidos to set
	 */
	public void setSubcontenidos(List<SubContenidoDTO> subcontenidos) {
		this.subcontenidos = subcontenidos;
	}
	/**
	 * @return the referencias
	 */
	@OneToMany( cascade = CascadeType.ALL)
	@JoinColumn(name="id_contenidos")
	public List<ReferenciaDTO> getReferencias() {
		return referencias;
	}
	/**
	 * @param referencias the referencias to set
	 */
```

```
	public void setReferencias(List<ReferenciaDTO> referencias) {
		this.referencias = referencias;
	}
	/**
	 * @return the temas_relacionados
	 */
	@OneToMany( cascade = CascadeType.ALL)
	@JoinColumn(name="id_contenidos")
	public List<TemaRelacionadoJoinDTO> getTemas_relacionados() {
		return temas_relacionados;
	}
	/**
	 * @param temas_relacionados the temas_relacionados to set
	 */
	public void setTemas_relacionados(List<TemaRelacionadoJoinDTO>
temas_relacionados) {
		this.temas_relacionados = temas_relacionados;
	}

}
```

8.3.7 ExamenJoinDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import org.hibernate.annotations.Proxy;
import javax.persistence.*;
import java.io.Serializable;
import java.util.List;

/**
 * Clase que almacena la información de un examen en el sistema
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name = "examenes")
@Proxy( lazy=false )
public class ExamenJoinDTO implements Serializable {

	/**
	 * id del examen, mapeado con el campo -id_examenes- de la tabla -examenes-
de la Base de Datos
	 */
	private long id_examenes;
	/**
	 * titulo del examen, mapeado con el campo -titulo- de la tabla -examenes- de
```

```
la Base de Datos
	 */
	private String titulo;
	/**
	 * status del examen, mapeado con el campo -status- de la tabla -examenes- de
la Base de Datos
	 */
	private int status;
	/**
	 * preguntas asociadas al examen, hace referencia a la tabla -preguntas- de
la Base de Datos, mapeada por el campo -id_examenes-
	 */
	private List<PreguntaDTO> preguntas;

	/**
	 * @return the id_examen
	 */
	@Id
	@Column(name="id_examenes")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId_examenes() {
		return id_examenes;
	}
	/**
	 * @return the titulo
	 */
	@Column(name="titulo")
	public String getTitulo() {
		return titulo;
	}
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @param id_examen the id_examen to set
	 */
	public void setId_examenes(long id_examenes) {
		this.id_examenes = id_examenes;
	}
	/**
	 * @param titulo the titulo to set
	 */
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}

	public void setStatus(int status) {
		this.status = status;
	}
	/**
	 * @return the preguntas
	 */
	@OneToMany( cascade = CascadeType.ALL)
```

```
	@JoinColumn(name="id_examenes")
	public List<PreguntaDTO> getPreguntas() {
		return preguntas;
	}
	/**
	 * @param preguntas the preguntas to set
	 */
	public void setPreguntas(List<PreguntaDTO> preguntas) {
		this.preguntas = preguntas;
	}
}
```

8.3.8 ExamenSaveDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import org.hibernate.annotations.Proxy;
import javax.persistence.*;
import java.io.Serializable;
import java.util.List;

/**
 * Clase que almacena la información de un examen en el sistema
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name = "examenes")
@Proxy( lazy=false )
public class ExamenSaveDTO implements Serializable {

	/**
	 * id del examen, mapeado con el campo -id_examenes- de la tabla -examenes-
de la Base de Datos
	 */
	private long id_examenes;
	/**
	 * titulo del examen, mapeado con el campo -titulo- de la tabla -examenes- de
la Base de Datos
	 */
	private String titulo;
	/**
	 * status del examen, mapeado con el campo -status- de la tabla -examenes- de
la Base de Datos
	 */
	private int status;
	/**
```

```
	 * preguntas asociadas al examen, hace referencia a la tabla -preguntas- de
la Base de Datos, mapeada por el campo -id_examenes-
	 */
	//private List<PreguntaDTO> preguntas;

	/**
	 * @return the id_examen
	 */
	@Id
	@Column(name="id_examenes")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId_examenes() {
		return id_examenes;
	}
	/**
	 * @return the titulo
	 */
	@Column(name="titulo")
	public String getTitulo() {
		return titulo;
	}
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @param id_examen the id_examen to set
	 */
	public void setId_examenes(long id_examenes) {
		this.id_examenes = id_examenes;
	}
	/**
	 * @param titulo the titulo to set
	 */
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}

	public void setStatus(int status) {
		this.status = status;
	}
	/**
	 * @return the preguntas
	 */
	//@OneToMany( cascade = CascadeType.ALL)
	//@JoinColumn(name="id_examenes")
	//public List<PreguntaDTO> getPreguntas() {
		//return preguntas;
	//}
	/**
	 * @param preguntas the preguntas to set
	 */
	//public void setPreguntas(List<PreguntaDTO> preguntas) {
		//this.preguntas = preguntas;
```

```
	//}
}
```

8.3.9 MedicoAdscritoJoinDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la información de un medico adscrito en el sistema
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name = "medicos_adscritos")
@PrimaryKeyJoinColumn(name="id")
@Proxy( lazy=false )
public class MedicoAdscritoJoinDTO extends UsuarioDTO implements Serializable{

	/**
	 * area del medico adscrito, mapeado con el campo -area- de la tabla
-medicos_adscritos- de la Base de Datos
	 */
	private String area;
	/**
	 * puesto del medico adscrito, mapeado con el campo -puesto- de la tabla
-medicos_adscritos- de la Base de Datos
	 */
	private String puesto;

	////////////////////////////// METODOS GETTERS Y
SETTERS ///////////////////////////////

	/**
	 * @return the area
	 */
	@Column(name="area")
	public String getArea() {
		return area;
```

```
	}
	/**
	 * @return the puesto
	 */
	@Column(name="puesto")
	public String getPuesto() {
		return puesto;
	}
	/**
	 * @param area the area to set
	 */
	public void setArea(String area) {
		this.area = area;
	}
	/**
	 * @param puesto the puesto to set
	 */
	public void setPuesto(String puesto) {
		this.puesto = puesto;
	}
}
```

8.3.10 MedicoAdscritoSaveDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la información de un medico adscrito en el sistema
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name = "medicos_adscritos")
@PrimaryKeyJoinColumn(name="id")
@Proxy( lazy=false )
public class MedicoAdscritoJoinDTO extends UsuarioDTO implements Serializable{

	/**
```

```
	 * area del medico adscrito, mapeado con el campo -area- de la tabla
-medicos_adscritos- de la Base de Datos
	 */
	private String area;
	/**
	 * puesto del medico adscrito, mapeado con el campo -puesto- de la tabla
-medicos_adscritos- de la Base de Datos
	 */
	private String puesto;

	////////////////////////// METODOS GETTERS Y
SETTERS //////////////////////////

	/**
	 * @return the area
	 */
	@Column(name="area")
	public String getArea() {
		return area;
	}
	/**
	 * @return the puesto
	 */
	@Column(name="puesto")
	public String getPuesto() {
		return puesto;
	}
	/**
	 * @param area the area to set
	 */
	public void setArea(String area) {
		this.area = area;
	}
	/**
	 * @param puesto the puesto to set
	 */
	public void setPuesto(String puesto) {
		this.puesto = puesto;
	}
}
```

8.3.11 MedicoResidenteJoinDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;
```

```java
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la información de un medico adscrito en el sistema
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name = "medicos_adscritos")
@PrimaryKeyJoinColumn(name="id")
@Proxy( lazy=false )
public class MedicoAdscritoJoinDTO extends UsuarioDTO implements Serializable{

	/**
	 * area del medico adscrito, mapeado con el campo -area- de la tabla
-medicos_adscritos- de la Base de Datos
	 */
	private String area;
	/**
	 * puesto del medico adscrito, mapeado con el campo -puesto- de la tabla
-medicos_adscritos- de la Base de Datos
	 */
	private String puesto;

	////////////////////////////// METODOS GETTERS Y
SETTERS ////////////////////////////

	/**
	 * @return the area
	 */
	@Column(name="area")
	public String getArea() {
		return area;
	}
	/**
	 * @return the puesto
	 */
	@Column(name="puesto")
	public String getPuesto() {
		return puesto;
	}
	/**
	 * @param area the area to set
	 */
	public void setArea(String area) {
		this.area = area;
	}
	/**
	 * @param puesto the puesto to set
```

```
	 */
	public void setPuesto(String puesto) {
		this.puesto = puesto;
	}
}
```

8.3.12 MedicoResidenteSaveDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLÁSIAAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
//import javax.persistence.FetchType;
//import javax.persistence.ManyToMany;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la informacion de un medico residente en el sistema, hereda de la clase UsuarioDTO
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name = "medicos_residentes")
@PrimaryKeyJoinColumn(name="id")
@Proxy( lazy=false )
public class MedicoResidenteSaveDTO extends UsuarioDTO implements Serializable{

	/**
	 * escuela del medico residente, mapeado con el campo -escuela- de la tabla -medicos_residentes- de la Base de Datos
	 */
	private String escuela;
	/**
	 * examenes asociados al medico residente, hace referencia a la tabla -examenes- de la Base de Datos, mapeada mediante la tabla intermedia -medicos_residentes_examenes- para persistir la relacion muchos a muchos
	 */
	//private List<ExamenDTO> examenes = new ArrayList<ExamenDTO>();
```

```
	//////////////////////////////// METODOS GETTERS Y
SETTERS ////////////////////////////////

	/**
	 * @return the escuela
	 */
	@Column(name="escuela")
	public String getEscuela() {
		return escuela;
	}

	/**
	 * @return the examen
	 */
	//@ManyToMany(cascade=CascadeType.ALL, fetch=FetchType.EAGER)
	//public List<ExamenDTO> getExamen() {
	//	return examenes;
	//}

	/**
	 * @param escuela the escuela to set
	 */
	public void setEscuela(String escuela) {
		this.escuela = escuela;
	}

	/**
	 * @param examen the examen to set
	 */
	/*public void setExamen(List<ExamenDTO> examenes) {
		this.examenes = examenes;
	}

	public void agregarExamen(ExamenDTO examen){
	this.examenes.add(examen);
    }*/
}
```

8.3.13 PacienteJoinDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
```

```
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la informacion de un paciente en el sistema, hereda de la
clase UsuarioDTO
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name="pacientes")
@PrimaryKeyJoinColumn(name="id")
@Proxy(lazy=false)
public class PacienteJoinDTO extends UsuarioDTO implements Serializable {

	/**
	 * numero de afiliacion del paciente, mapeado con el campo -no_afiliacion- de
la tabla -pacientes- de la Base de Datos
	 */
	private String no_afiliacion;
	/**
	 * tipo de sasngre del paciente, mapeado con el campo -tipo_sangre- de la
tabla -pacientes- de la Base de Datos
	 */
	private String tipo_sangre;
	/**
	 * alergias del paciente, mapeado con el campo -alergias- de la tabla
-pacientes- de la Base de Datos
	 */
	private String alergias;
	/**
	 * cuidados del paciente, mapeado con el campo -cuidados- de la tabla
-pacientes- de la Base de Datos
	 */
	private String cuidados;
	/**
	 * enfermedades del paciente, mapeado con el campo -enfermedades- de la
tabla -pacientes- de la Base de Datos
	 */
	private String enfermedades;

	////////////////////////////////// METODOS GETTERS Y
SETTERS //////////////////////////////////

	/**
	 * @return the no_afiliacion
	 */
	@Column(name="no_afiliacion")
	public String getNo_afiliacion() {
		return no_afiliacion;
	}
	/**
```

```
	 * @return the tipo_sangre
	 */
	@Column(name="tipo_sangre")
	public String getTipo_sangre() {
		return tipo_sangre;
	}
	/**
	 * @return the alergias
	 */
	@Column(name="alergias")
	public String getAlergias() {
		return alergias;
	}
	/**
	 * @return the cuidados
	 */
	@Column(name="cuidados")
	public String getCuidados() {
		return cuidados;
	}
	/**
	 * @return the enfermedades
	 */
	@Column(name="enfermedades")
	public String getEnfermedades() {
		return enfermedades;
	}

	/**
	 * @param no_afiliacion the no_afiliacion to set
	 */
	public void setNo_afiliacion(String no_afiliacion) {
		this.no_afiliacion = no_afiliacion;
	}
	/**
	 * @param tipo_sangre the tipo_sangre to set
	 */
	public void setTipo_sangre(String tipo_sangre) {
		this.tipo_sangre = tipo_sangre;
	}
	/**
	 * @param alergias the alergias to set
	 */
	public void setAlergias(String alergias) {
		this.alergias = alergias;
	}
	/**
	 * @param cuidados the cuidados to set
	 */
	public void setCuidados(String cuidados) {
		this.cuidados = cuidados;
	}
	/**
	 * @param enfermedades the enfermedades to set
```

```
	 */
	public void setEnfermedades(String enfermedades) {
		this.enfermedades = enfermedades;
	}
}
```

8.3.14 PacienteSaveDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la informacion de un paciente en el sistema, hereda de la clase UsuarioDTO
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name="pacientes")
@PrimaryKeyJoinColumn(name="id")
@Proxy(lazy=false)
public class PacienteSaveDTO extends UsuarioDTO implements Serializable {

	/**
	 * numero de afiliacion del paciente, mapeado con el campo -no_afiliacion- de la tabla -pacientes- de la Base de Datos
	 */
	private String no_afiliacion;
	/**
	 * tipo de sasngre del paciente, mapeado con el campo -tipo_sangre- de la tabla -pacientes- de la Base de Datos
	 */
	private String tipo_sangre;
	/**
	 * alergias del paciente, mapeado con el campo -alergias- de la tabla -pacientes- de la Base de Datos
	 */
	private String alergias;
```

```
	/**
	 * cuidados del paciente, mapeado con el campo -cuidados- de la tabla
-pacientes- de la Base de Datos
	 */
	private String cuidados;
	/**
	 * enfermedades del paciente, mapeado con el campo -enfermedades- de la
tabla -pacientes- de la Base de Datos
	 */
	private String enfermedades;

	////////////////////////////////// METODOS GETTERS Y
SETTERS //////////////////////////////////

	/**
	 * @return the no_afiliacion
	 */
	@Column(name="no_afiliacion")
	public String getNo_afiliacion() {
		return no_afiliacion;
	}
	/**
	 * @return the tipo_sangre
	 */
	@Column(name="tipo_sangre")
	public String getTipo_sangre() {
		return tipo_sangre;
	}
	/**
	 * @return the alergias
	 */
	@Column(name="alergias")
	public String getAlergias() {
		return alergias;
	}
	/**
	 * @return the cuidados
	 */
	@Column(name="cuidados")
	public String getCuidados() {
		return cuidados;
	}
	/**
	 * @return the enfermedades
	 */
	@Column(name="enfermedades")
	public String getEnfermedades() {
		return enfermedades;
	}

	/**
	 * @param no_afiliacion the no_afiliacion to set
	 */
	public void setNo_afiliacion(String no_afiliacion) {
```

```
		this.no_afiliacion = no_afiliacion;
	}
	/**
	 * @param tipo_sangre the tipo_sangre to set
	 */
	public void setTipo_sangre(String tipo_sangre) {
		this.tipo_sangre = tipo_sangre;
	}
	/**
	 * @param alergias the alergias to set
	 */
	public void setAlergias(String alergias) {
		this.alergias = alergias;
	}
	/**
	 * @param cuidados the cuidados to set
	 */
	public void setCuidados(String cuidados) {
		this.cuidados = cuidados;
	}
	/**
	 * @param enfermedades the enfermedades to set
	 */
	public void setEnfermedades(String enfermedades) {
		this.enfermedades = enfermedades;
	}
}
```

8.3.15 PreguntaDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import org.hibernate.annotations.Proxy;
import javax.persistence.*;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;


/**
 * Clase que almacena la informacion de una pregunta en el sistema
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity @Table(name="preguntas")
@Proxy (lazy = false)
```

```java
public class PreguntaDTO implements Serializable {

	/**
	 * id de la pregunta, mapeado con el campo -id_preguntas- de la tabla
-preguntas- de la Base de Datos
	 */
	private long id_preguntas;
	/**
	 * pregunta en si de la pregunta, mapeado con el campo -pregunta- de la
tabla -preguntas- de la Base de Datos
	 */
	private String pregunta;
	/**
	 * status de la pregunta, mapeado con el campo -status- de la tabla
-preguntas- de la Base de Datos
	 */
	private int status;
	/**
	 * respuestas asociadas a la pregunta, hace referencia a la tabla
-respuestas- de la Base de Datos, mapeada por el campo -id_preguntas-
	 */
	private List<RespuestaDTO> respuestas;

	////////////////////////////////////////////////////////////////////////
///
	/**
	 * @return the id_examenes
	 */
	@Id @Column(name="id_preguntas")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId_preguntas() {
		return id_preguntas;
	}
	/**
	 * @return the pregunta
	 */
	@Column(name="pregunta")
	public String getPregunta() {
		return pregunta;
	}
	/**
	 * @return the status
	 */
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @return the respuestas
	 */
	@OneToMany( cascade = CascadeType.ALL)
	@JoinColumn(name="id_preguntas")
	public List<RespuestaDTO> getRespuestas() {
		return respuestas;
```

```
	}

	////////////////////////////////////////////////////////////////////////////
////

	/**
	 * @param id_examenes the id_examenes to set
	 */
	public void setId_preguntas(long id_preguntas) {
		this.id_preguntas = id_preguntas;
	}
	/**
	 * @param pregunta the pregunta to set
	 */
	public void setPregunta(String pregunta) {
		this.pregunta = pregunta;
	}
	/**
	 * @param status the status to set
	 */
	public void setStatus(int status) {
		this.status = status;
	}
	/**
	 * @param respuestas the respuestas to set
	 */
	public void setRespuestas(List<RespuestaDTO> respuestas) {
		this.respuestas = respuestas;
	}

}
```

8.3.16 ReferenciaDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;
```

```java
/**
 * Clase que almacena la informacion de una referencia en el sistema asociada a un contenido, de la clase ContenidoDTO
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name="referencias")
@Proxy(lazy=false)
public class ReferenciaDTO implements Serializable {

	/**
	 * id de la referencia, mapeado con el campo -id_referencias- de la tabla -referencias- de la Base de Datos
	 */
	private long id_referencias;
	/**
	 * referencia en si de la referencia, mapeado con el campo -referencia- de la tabla -referencias- de la Base de Datos
	 */
	private String referencia;
	/**
	 * link de la referencia, mapeado con el campo -link- de la tabla -referencias- de la Base de Datos
	 */
	private String link;
	/**
	 * status de la referencia, mapeado con el campo -status- de la tabla -referencias- de la Base de Datos
	 */
	private int status;

	/**
	 * @return the status
	 */
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @param status the status to set
	 */
	public void setStatus(int status) {
		this.status = status;
	}
	/**
	 * @return the id_referencias
	 */
	@Id
	@Column(name="id_referencias")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId_referencias() {
		return id_referencias;
	}
```

```
	/**
	 * @return the referencia
	 */
	@Column(name="referencia")
	public String getReferencia() {
		return referencia;
	}
	/**
	 * @return the link
	 */
	@Column(name="link")
	public String getLink() {
		return link;
	}
	/**
	 * @param id_referencias the id_referencias to set
	 */
	public void setId_referencias(long id_referencias) {
		this.id_referencias = id_referencias;
	}
	/**
	 * @param referencia the referencia to set
	 */
	public void setReferencia(String referencia) {
		this.referencia = referencia;
	}
	/**
	 * @param link the link to set
	 */
	public void setLink(String link) {
		this.link = link;
	}

}
```

8.3.17 RespuestaDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
```

```java
import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la informacion de una respuesta en el sistema, asociada a una pregunta de la clase PreguntaDTO
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table (name="respuestas")
@Proxy (lazy=false)
public class RespuestaDTO implements Serializable {

	/**
	 * id de la respuesta, mapeado con el campo -id_respuestas- de la tabla -respuestas- de la Base de Datos
	 */
	private long id_respuestas;
	/**
	 * respuesta en si de la respuesta, mapeado con el campo -respuesta- de la tabla -respuestas- de la Base de Datos
	 */
	private String respuesta;
	/**
	 * veracidad de la respuesta, mapeado con el campo -correcta- de la tabla -respuestas- de la Base de Datos
	 */
	private int correcta;
	/**
	 * status de la respuesta, mapeado con el campo -status- de la tabla -respuestas- de la Base de Datos
	 */
	private int status;

	////////////////////////////////////////////////////////////////////////////

	/**
	 * @return the id_respuestas
	 */
	@Column
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId_respuestas() {
		return id_respuestas;
	}
	/**
	 * @return the respuesta
	 */
	@Column(name="respuesta")
	public String getRespuesta() {
		return respuesta;
	}
```

```
	/**
	 * @return the correcta
	 */
	@Column(name="correcta")
	public int getCorrecta() {
		return correcta;
	}
	/**
	 * @return the status
	 */
	@Column(name="status")
	public int getStatus() {
		return status;
	}

	//////////////////////////////// METODOS
SETTERS ////////////////////////////

	/**
	 * @param id_respuestas the id_respuestas to set
	 */
	public void setId_respuestas(long id_respuestas) {
		this.id_respuestas = id_respuestas;
	}
	/**
	 * @param respuesta the respuesta to set
	 */
	public void setRespuesta(String respuesta) {
		this.respuesta = respuesta;
	}
	/**
	 * @param correcta the correcta to set
	 */
	public void setCorrecta(int correcta) {
		this.correcta = correcta;
	}
	/**
	 * @param status the status to set
	 */
	public void setStatus(int status) {
		this.status = status;
	}

}
```

8.3.18 SubContenidoDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;
```

```java
import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la informacion de un subcontenido en el sistema, asociado a un contenido de la clase ContenidoDTO
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name="subcontenidos")
@Proxy(lazy=false)
public class SubContenidoDTO implements Serializable {

	/**
	 * id del subcontenido, mapeado con el campo -id_subcontenidos- de la tabla -subcontenidos- de la Base de Datos
	 */
	private long id_sucontenidos;
	/**
	 * descripcion del subcontenido, mapeado con el campo -descripcion_subcontenido- de la tabla -subcontenidos- de la Base de Datos
	 */
	private String desripcion_subcontenidos;
	/**
	 * status del subcontenido, mapeado con el campo -status- de la tabla -subcontenidos- de la Base de Datos
	 */
	private int status;

	////////////////////////////////////////////////////////////////////////////////

	/**
	 * @return the status
	 */
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @param status the status to set
	 */
	public void setStatus(int status) {
		this.status = status;
	}
```

```
	/**
	 * @return the id_sucontenidos
	 */
	@Id
	@Column(name="id_subcontenidos")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId_sucontenidos() {
		return id_sucontenidos;
	}
	/**
	 * @return the desripcion_subcontenidos
	 */
	@Column(name="descripcion_subcontenido")
	public String getDesripcion_subcontenidos() {
		return desripcion_subcontenidos;
	}
	/**
	 * @param id_sucontenidos the id_sucontenidos to set
	 */
	public void setId_sucontenidos(long id_sucontenidos) {
		this.id_sucontenidos = id_sucontenidos;
	}
	/**
	 * @param desripcion_subcontenidos the desripcion_subcontenidos to set
	 */
	public void setDesripcion_subcontenidos(String desripcion_subcontenidos) {
		this.desripcion_subcontenidos = desripcion_subcontenidos;
	}
}
```

8.3.19 TemaRelacionadoJoinDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la informacion de un tema relacionado en el sistema,
asociado a un contenido de la clase ContenidoDTO
 * @version 1.0, 25/04/2012
```

```
 * @author Roberto Gonzalez
 */
@Entity
@Table(name="temas_relacionados")
@Proxy(lazy=false)
public class TemaRelacionadoJoinDTO implements Serializable {

	/**
	 * id del tema relacionado, mapeado con el campo -id_tema_relacionado- de la tabla -temas_relacionados- de la Base de Datos
	 */
	private long id_temas_relacionados;
	/**
	 * titulo del tema relacionado, mapeado con el campo -tema_relacionado- de la tabla -temas_relacionados- de la Base de Datos
	 */
	private String tema_relacionado;
	/**
	 * link del tema relacionado, mapeado con el campo -link- de la tabla -temas_relacionados- de la Base de Datos
	 */
	private String link;
	/**
	 * status del tema relacionado, mapeado con el campo -status- de la tabla -temas_relacionados- de la Base de Datos
	 */
	private int status;

	/**
	 * @return the id_temas_relacionados
	 */
	@Id
	@Column(name="id_temas_relacionados")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId_temas_relacionados() {
		return id_temas_relacionados;
	}
	/**
	 * @return the tema_relacionado
	 */
	@Column(name="tema_relacionado")
	public String getTema_relacionado() {
		return tema_relacionado;
	}
	/**
	 * @return the link
	 */
	@Column(name="link")
	public String getLink() {
		return link;
	}
	/**
	 * @return the status
	 */
```

```
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @param id_temas_relacionados the id_temas_relacionados to set
	 */
	public void setId_temas_relacionados(long id_temas_relacionados) {
		this.id_temas_relacionados = id_temas_relacionados;
	}
	/**
	 * @param tema_relacionado the tema_relacionado to set
	 */
	public void setTema_relacionado(String tema_relacionado) {
		this.tema_relacionado = tema_relacionado;
	}
	/**
	 * @param link the link to set
	 */
	public void setLink(String link) {
		this.link = link;
	}
	/**
	 * @param status the status to set
	 */
	public void setStatus(int status) {
		this.status = status;
	}

}
```

8.3.20 TemaRelacionadoSaveDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import org.hibernate.annotations.Proxy;

/**
```

```java
 * Clase que almacena la informacion de un tema relacionado en el sistema,
asociado a un contenido de la clase ContenidoDTO
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name="temas_relacionados")
@Proxy(lazy=false)
public class TemaRelacionadoSaveDTO implements Serializable {

	/**
	 * id del tema relacionado, mapeado con el campo -id_tema_relacionado- de la
tabla -temas_relacionados- de la Base de Datos
	 */
	private long id_temas_relacionados;
	/**
	 * titulo del tema relacionado, mapeado con el campo -tema_relacionado- de la
tabla -temas_relacionados- de la Base de Datos
	 */
	private String tema_relacionado;
	/**
	 * link del tema relacionado, mapeado con el campo -link- de la tabla
-temas_relacionados- de la Base de Datos
	 */
	private String link;
	/**
	 * status del tema relacionado, mapeado con el campo -status- de la tabla
-temas_relacionados- de la Base de Datos
	 */
	private int status;

	/**
	 * @return the id_temas_relacionados
	 */
	@Id
	@Column(name="id_temas_relacionados")
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	public long getId_temas_relacionados() {
		return id_temas_relacionados;
	}
	/**
	 * @return the tema_relacionado
	 */
	@Column(name="tema_relacionado")
	public String getTema_relacionado() {
		return tema_relacionado;
	}
	/**
	 * @return the link
	 */
	@Column(name="link")
	public String getLink() {
		return link;
	}
```

```
	/**
	 * @return the status
	 */
	@Column(name="status")
	public int getStatus() {
		return status;
	}
	/**
	 * @param id_temas_relacionados the id_temas_relacionados to set
	 */
	public void setId_temas_relacionados(long id_temas_relacionados) {
		this.id_temas_relacionados = id_temas_relacionados;
	}
	/**
	 * @param tema_relacionado the tema_relacionado to set
	 */
	public void setTema_relacionado(String tema_relacionado) {
		this.tema_relacionado = tema_relacionado;
	}
	/**
	 * @param link the link to set
	 */
	public void setLink(String link) {
		this.link = link;
	}
	/**
	 * @param status the status to set
	 */
	public void setStatus(int status) {
		this.status = status;
	}

}
```

8.3.21 UsuarioDTO

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.data;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.ManyToOne;
```

```
import javax.persistence.Table;
import javax.persistence.InheritanceType;

import org.hibernate.annotations.Proxy;

/**
 * Clase que almacena la informacion de un usuario en el sistema
 * @version 1.0, 25/04/2012
 * @author Roberto Gonzalez
 */
@Entity
@Table(name = "usuarios")
@Inheritance(strategy=InheritanceType.JOINED)
@Proxy( lazy=false )
public class UsuarioDTO implements Serializable {

	/**
	 * id del usuario, mapeado con el campo -id- de la tabla -usuarios- de la
Base de Datos
	 */
	private long id;
	/**
	 * login del usuario, mapeado con el campo -login- de la tabla -usuarios- de
la Base de Datos
	 */
	private String login;
	/**
	 * password del usuario, mapeado con el campo -password- de la tabla
-usuarios- de la Base de Datos
	 */
	private String password;
	/**
	 * nombres del usuario, mapeado con el campo -nombres- de la tabla -usuarios-
de la Base de Datos
	 */
	private String nombres;
	/**
	 * apellido paterno del usuario, mapeado con el campo -ap_paterno- de la
tabla -usuarios- de la Base de Datos
	 */
	private String ap_paterno;
	/**
	 * apellido materno del usuario, mapeado con el campo -ap_materno- de la
tabla -usuarios- de la Base de Datos
	 */
	private String ap_materno;
	/**
	 * telefono de casa del usuario, mapeado con el campo -tel_casa- de la tabla
-usuarios- de la Base de Datos
	 */
	private String tel_casa;
	/**
	 * telefono de trabajo del usuario, mapeado con el campo -tel_trabajo- de la
tabla -usuarios- de la Base de Datos
```

```
	 */
	private String tel_trabajo;
	/**
	 * fax del usuario, mapeado con el campo -fax- de la tabla -usuarios- de la
Base de Datos
	 */
	private String fax;
	/**
	 * celular del usuario, mapeado con el campo -celular- de la tabla
-usuarios- de la Base de Datos
	 */
	private String celular;
	/**
	 * direccion del usuario, mapeado con el campo -direccion- de la tabla
-usuarios- de la Base de Datos
	 */
	private String direccion;
	/**
	 * direccion dos del usuario, mapeado con el campo -direccion_2- de la tabla
-usuarios- de la Base de Datos
	 */
	private String direccion2;
	/**
	 * codigo postal del usuario, mapeado con el campo -codigo_postal- de la
tabla -usuarios- de la Base de Datos
	 */
	private String codigo_postal;
	/**
	 * email del usuario, mapeado con el campo -email- de la tabla -usuarios- de
la Base de Datos
	 */
	private String email;
	/**
	 * genero del usuario, mapeado con el campo -genero- de la tabla -usuarios-
de la Base de Datos
	 */
	private char genero;
	/**
	 * fecha de nacimiento del usuario, mapeado con el campo -fecha_nacimiento-
de la tabla -usuarios- de la Base de Datos
	 */
	private Date fecha_nacimiento;
	/**
	 * fecha de ingreso al sistema del usuario, mapeado con el campo
-fecha_ingreso- de la tabla -usuarios- de la Base de Datos
	 */
	private Date fecha_ingreso;
	/**
	 * codigo de pais del usuario, mapeado con el campo -Country_Code- de la
tabla -usuarios- de la Base de Datos
	 */
	private String pais;
	/**
	 * perfil del usuario, mapeado con el campo -perfil- de la tabla -usuarios-
```

```
de la Base de Datos
	 */
	private int perfil;
	/**
	 * status del usuario, mapeado con el campo -status- de la tabla -usuarios-
de la Base de Datos
	 */
	private int status;

	////////////////////////////////// METODOS GETTERS Y
SETTERS ////////////////////////////////

	/**
	 * @return the id
	 */
	@Id
    @GeneratedValue
    @Column(name = "id", unique = true, nullable = false)
	public long getId() {
		return id;
	}
	/**
	 * @return the login
	 */
	@Column(name = "login")
	public String getLogin() {
		return login;
	}
	/**
	 * @return the password
	 */
	@Column(name = "password")
	public String getPassword() {
		return password;
	}
	/**
	 * @return the nombres
	 */
	@Column(name = "nombres")
	public String getNombres() {
		return nombres;
	}
	/**
	 * @return the apPaterno
	 */
	@Column(name = "ap_paterno")
	public String getAp_paterno() {
		return ap_paterno;
	}
	/**
	 * @return the apMaterno
	 */
	@Column(name = "ap_materno")
	public String getAp_materno() {
```

```
        return ap_materno;
    }
    /**
     * @return the telCasa
     */
    @Column(name = "tel_casa")
    public String getTel_casa() {
        return tel_casa;
    }
    /**
     * @return the telTrabajo
     */
    @Column(name = "tel_trabajo")
    public String getTel_trabajo() {
        return tel_trabajo;
    }
    /**
     * @return the fax
     */
    @Column(name = "fax")
    public String getFax() {
        return fax;
    }
    /**
     * @return the celular
     */
    @Column(name = "celular")
    public String getCelular() {
        return celular;
    }
    /**
     * @return the direccion
     */
    @Column(name = "direccion")
    public String getDireccion() {
        return direccion;
    }
    /**
     * @return the direccion2
     */
    @Column(name = "direccion_2")
    public String getDireccion2() {
        return direccion2;
    }
    /**
     * @return the codigoPostal
     */
    @Column(name = "codigo_postal")
    public String getCodigo_postal() {
        return codigo_postal;
    }
    /**
     * @return the email
     */
```

```java
@Column(name = "email")
public String getEmail() {
	return email;
}
/**
 * @return the genero
 */
@Column(name = "genero")
public char getGenero() {
	return genero;
}
/**
 * @return the fechaNacimiento
 */
@Column(name = "fecha_nacimiento")
public Date getFecha_nacimiento() {
	return fecha_nacimiento;
}
/**
 * @return the fechaIngreso
 */
@Column(name = "fecha_ingreso")
public Date getFecha_ingreso() {
	return fecha_ingreso;
}
/**
 * @return the countryCode
 */
@Column(name="pais")
public String getPais() {
	return pais;
}
/**
 * @return the perfil
 */
@Column(name="perfil")
public int getPerfil() {
	return perfil;
}
@Column(name="status")
public int getStatus() {
	return status;
}
/**
 * @param id the id to set
 */
public void setId(long id) {
	this.id = id;
}
/**
 * @param login the login to set
 */
public void setLogin(String login) {
	this.login = login;
```

```
}
/**
 * @param password the password to set
 */
public void setPassword(String password) {
	this.password = password;
}
/**
 * @param nombres the nombres to set
 */
public void setNombres(String nombres) {
	this.nombres = nombres;
}
/**
 * @param apPaterno the apPaterno to set
 */
public void setAp_paterno(String apPaterno) {
	this.ap_paterno = apPaterno;
}
/**
 * @param apMaterno the apMaterno to set
 */
public void setAp_materno(String apMaterno) {
	this.ap_materno = apMaterno;
}
/**
 * @param telCasa the telCasa to set
 */
public void setTel_casa(String telCasa) {
	this.tel_casa = telCasa;
}
/**
 * @param telTrabajo the telTrabajo to set
 */
public void setTel_trabajo(String telTrabajo) {
	this.tel_trabajo = telTrabajo;
}
/**
 * @param fax the fax to set
 */
public void setFax(String fax) {
	this.fax = fax;
}
/**
 * @param celular the celular to set
 */
public void setCelular(String celular) {
	this.celular = celular;
}
/**
 * @param direccion the direccion to set
 */
public void setDireccion(String direccion) {
	this.direccion = direccion;
```

```
	}
	/**
	 * @param direccion2 the direccion2 to set
	 */
	public void setDireccion2(String direccion2) {
		this.direccion2 = direccion2;
	}
	/**
	 * @param codigoPostal the codigoPostal to set
	 */
	public void setCodigo_postal(String codigoPostal) {
		this.codigo_postal = codigoPostal;
	}
	/**
	 * @param email the email to set
	 */
	public void setEmail(String email) {
		this.email = email;
	}
	/**
	 * @param genero the genero to set
	 */
	public void setGenero(char genero) {
		this.genero = genero;
	}
	/**
	 * @param fechaNacimiento the fechaNacimiento to set
	 */
	public void setFecha_nacimiento(Date fechaNacimiento) {
		this.fecha_nacimiento = fechaNacimiento;
	}
	/**
	 * @param fechaIngreso the fechaIngreso to set
	 */
	public void setFecha_ingreso(Date fechaIngreso) {
		this.fecha_ingreso = fechaIngreso;
	}
	/**
	 * @param countryCode the countryCode to set
	 */
	public void setPais(String pais) {
		this.pais = pais;
	}
	/**
	 * @param perfil the perfil to set
	 */
	public void setPerfil(int perfil) {
		this.perfil = perfil;
	}
	public void setStatus(int status) {
		this.status = status;
	}
}
```

8.4 mx.uam.azc.pt.test

8.4.1 TestRoberto

```
/**
*  PROTOTIPO DE SISTEMA DE APRENDIZAJE PARA MÉDICOS RESIDENTES
*  EN EL SEGUIMIENTO DEL TRATAMIENTO DE ENFERMEDADES NEOPLASICAS
*/
package mx.uam.azc.pt.test;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import mx.uam.azc.pt.business.AdministradorAdministradores;
import mx.uam.azc.pt.business.AdministradorCasos;
import mx.uam.azc.pt.business.AdministradorContenidos;
import mx.uam.azc.pt.business.AdministradorExamenes;
import mx.uam.azc.pt.business.AdministradorMedicosAdscritos;
import mx.uam.azc.pt.business.AdministradorMedicosResidentes;
import mx.uam.azc.pt.business.AdministradorPacientes;
import mx.uam.azc.pt.business.AdministradorPreguntas;
import mx.uam.azc.pt.data.AdministradorDTO;
import mx.uam.azc.pt.data.CasoDTO;
import mx.uam.azc.pt.data.ContenidoDTO;
import mx.uam.azc.pt.data.ExamenDTO;
import mx.uam.azc.pt.data.MedicoAdscritoDTO;
import mx.uam.azc.pt.data.MedicoResidenteDTO;
import mx.uam.azc.pt.data.PacienteDTO;
import mx.uam.azc.pt.data.PreguntaDTO;
import mx.uam.azc.pt.data.ReferenciaDTO;
import mx.uam.azc.pt.data.RespuestaDTO;
import mx.uam.azc.pt.data.SubContenidoDTO;
import mx.uam.azc.pt.data.TemaRelacionadoDTO;

/**
 * @author Roberto Gonzalez
 * clase para probar inserciones, consultas, bajas y cambios en cuanto al manejo
de persistencia
 */
public class TestRoberto {
```

```
	/**
	 * @param args
	 * metodo main de java necesario para ejecutar una clase
	 */
	public static void main(String[] args) {
		try
		{
		  new TestRoberto().execute();
		}
		catch ( Exception e )
		{
		   e.printStackTrace() ;
		}
	}

	/**
	 * metodo execute, se ejecuta para probar el manejo de persistencia, puede lanzar excepciones
	 */
	private void execute() throws Exception
	  {
		System.out.println("empiezo");
	    //pruebaHerencia();
		  //pruebaLectura();
		  //pruebaInsercionMEdico Adscrito();
		  //pruebaInsercionAdministrador();
		  //pruebaInsercionPaciente();
		  //pruebaActualizacionPaciente();
		  //pruebaLecturaPaciente();
		  //pruebaLecturayActualizacionMedicoResidente();
		  //pruebaLecturayActualizacionMedicosAdscritos();
		  //pruebaLecturayActualizacionAdministrador();
		  //pruebaLecturayActualizacionPacientes();
		  //pruebaEliminacionCatalogos();
		  //pruebaInsertarExamen();
		  //pruebaAsignacionExamen();
		  //pruebaLecturaMedicosResidentesTodos();
		  //pruebaLecturaAdministradoresTodos();
		  //pruebaLecturaMedicosAdscritosTodos();
		  //pruebaLecturaPacientesTodos();
		  //pruebaExamenesConPreguntas();
		  //pruebaPreguntasConRespuestas();
		  //pruebaExamenConPreguntasYRespuestas();
		  //pruebaInsercionCasos();
		  //pruebaLecturaYActualizacionCasos();
		  //pruebaInsercionContenidos();
		  //pruebaLecturayActualizacionContenidos();
		  //pruebaLecturaContenidosTodos();
		  //pruebaInsercionContenidosConSubcontenidos();
		  //pruebaInsercionContenidosConReferencias();
		    pruebaInsercionContenidosConTemasRelacionados();
	    System.out.println("acabo");
	  }
```

```
	/**
	 * metodo que prueba la insercion de un contenido a la base de datos con
todo y temas relacionados
	 */
	public void pruebaInsercionContenidosConTemasRelacionados(){

		AdministradorContenidos administrador = getAdministradorContenidos();

		ContenidoDTO contenido = new ContenidoDTO();
		contenido=administrador.leerContenido(1);

		TemaRelacionadoDTO tema = new TemaRelacionadoDTO();
		tema.setTema_relacionado("neoplasias");
		tema.setLink("http://www.neoplasias.com.mx");

		List<TemaRelacionadoDTO> temas = new ArrayList<TemaRelacionadoDTO>();

		temas.add(tema);

		contenido.setTemas_relacionados(temas);

		administrador.actualizarContenido(contenido);
	}

	/**
	 * metodo que prueba la insercion de un contenido con todo y referencias a
la base de datos
	 */
	public void pruebaInsercionContenidosConReferencias(){

		AdministradorContenidos administrador = getAdministradorContenidos();

		ReferenciaDTO referencia = new ReferenciaDTO();

		referencia.setReferencia("link a hematomas tema");
		referencia.setLink("http://www.google.com.mx/juarjuar");

		List<ReferenciaDTO> referencias = new ArrayList<ReferenciaDTO>();

		referencias.add(referencia);

		ContenidoDTO contenido = administrador.leerContenido(1);

		contenido.setReferencias(referencias);

		administrador.actualizarContenido(contenido);

	}

	/**
	 * metodo que prueba la insercion de un contenido con todo y subcontenidos a
la base de datos
	 */
```

```
	public void pruebaInsercionContenidosConSubcontenidos(){


		AdministradorContenidos administrador = getAdministradorContenidos();

		ContenidoDTO contenido = new ContenidoDTO();
		contenido=administrador.leerContenido(1);
		System.out.println("ID: " + contenido.getId_contenidos());

		SubContenidoDTO sub = administrador.leerSubContenido(1);
		System.out.println("IDs: " + sub.getId_sucontenidos());

		sub.setDesripcion_subcontenidos("agdfsgaerg");

		List<SubContenidoDTO> subcontenidos = new ArrayList<SubContenidoDTO>();

		subcontenidos.add(sub);

		contenido.setSubcontenidos(subcontenidos);

		administrador.actualizarContenido(contenido);

	}

	/**
	 * metodo que prueba la lectura de todos los contenidos de la base de datos
	 */
	public void pruebaLecturaContenidosTodos(){

		AdministradorContenidos administrador = getAdministradorContenidos();

		List<ContenidoDTO> contenidos = new ArrayList<ContenidoDTO>();

		contenidos = administrador.leerContenidos();

		System.out.println("tamaño lista: " + contenidos.size());

	}

	/**
	 * metodo que prueba la lectura y la actualizacion de un contenido de la
base de datos
	 */
	public void pruebaLecturayActualizacionContenidos(){

		AdministradorContenidos administrador = getAdministradorContenidos();

		ContenidoDTO contenido = new ContenidoDTO();
		contenido = administrador.leerContenido(1);
		System.out.println(contenido.getTitulo_contenido());

		contenido.setTitulo_contenido("Hematomas bla bla");

		administrador.actualizarContenido(contenido);
```

```
	}

	/**
	 * metodo que prueba la insercion de un contenido
	 */
	public void pruebaInsercionContenidos(){

		AdministradorContenidos administrador = getAdministradorContenidos();

		ContenidoDTO contenido = new ContenidoDTO();

		contenido.setTitulo_contenido("Neoplasias");
		contenido.setDescripcion("Contenido neoplasias contenidos neoplasias
contenido neoplasias contenidos neoplasias");
		//contenido.setStatus(0);
		administrador.insertarContenido(contenido);

	}

	/**
	 * metodo que prueba la lectura y la actualizacion de un caso de la base de
datos
	 */
	public void pruebaLecturaYActualizacionCasos()
	{
		AdministradorCasos administrador = getAdministradorCasos();

		CasoDTO caso = new CasoDTO();

		caso = administrador.leerCaso(1);
		System.out.println("Caso: " + caso.getTitulo());

		caso.setTitulo("Hematomas adulto mayor");

		administrador.actualizarCaso(caso);
	}

	/**
	 * metodo que prueba la insercion de un caso a la base de datos
	 */
	public void pruebaInsercionCasos(){

		AdministradorCasos administrador = getAdministradorCasos();

		CasoDTO caso = new CasoDTO();

		caso.setTitulo("Neoplasias infantil");
		caso.setContenido_caso("Paciente de 73 años de edad valorado en la
consulta de Neurología de forma periódica a lo largo de 2 años y diagnosticado de
enfermedad de Parkinson por una clínica cardinal consistente en temblor de reposo
de predominio en miembro superior derecho, rigidez, bradicinesia y alteración de
reflejos posturales. Seguía tratamiento con levodopa y selegilina con adecuado
control de sus síntomas.");
```

```
            administrador.insertarCaso(caso);
      }

      /**
       * metodo que prueba la insercion de un examen con todo y preguntas y las
preguntas con todo y respuestas a la base de datos
       */
      public void pruebaExamenConPreguntasYRespuestas(){

            AdministradorExamenes administrador = getAdministradorExamenes();

            ExamenDTO examen = new ExamenDTO();

            examen.setTitulo("Diseño de Algoritmos");
            examen.setStatus(1);

            PreguntaDTO pregunta = new PreguntaDTO();

            pregunta.setPregunta("¿Que es la programación dinámica?");
            pregunta.setStatus(1);

            RespuestaDTO respuesta = new RespuestaDTO();

            respuesta.setRespuesta(" método para reducir el tiempo de ejecución de
un algoritmo mediante la utilización de subproblemas");
            respuesta.setStatus(1);
            respuesta.setCorrecta(1);

            List<PreguntaDTO> preguntas = new ArrayList<PreguntaDTO>();
            preguntas.add(pregunta);

            List<RespuestaDTO> respuestas = new ArrayList<RespuestaDTO>();
            respuestas.add(respuesta);

            pregunta.setRespuestas(respuestas);
            examen.setPreguntas(preguntas);

            administrador.insertarExamen(examen);

      }

      /**
       * metoodo que prueba la lectura y actualizacion de de una pregunta con todo
y respuestas de la base de datos
       */
      public void pruebaPreguntasConRespuestas(){

            AdministradorExamenes administrador = getAdministradorExamenes();

            PreguntaDTO pregunta = new PreguntaDTO();
            pregunta = administrador.leerPregunta(1);

            pregunta.setPregunta("¿Bajo que plataformas se puede compilar y
```

```
ejecutar Java?");

            List<RespuestaDTO> respuestas = new ArrayList<RespuestaDTO>();
            RespuestaDTO respuesta1 = administrador.leerRespuesta(1);
            respuesta1.setCorrecta(1);
            respuesta1.setRespuesta("No exxiste");
            respuesta1.setStatus(1);

            RespuestaDTO respuesta2 = administrador.leerRespuesta(2);
            respuesta2.setCorrecta(0);
            respuesta2.setRespuesta("solo bajo Linux");
            respuesta2.setStatus(1);

            respuestas.add(respuesta1);
            respuestas.add(respuesta2);

            pregunta.setRespuestas(respuestas);

            administrador.actualizarPregunta(pregunta);

      }

      /**
       * metodo que prueba la lectura y actualizacion de un examen con todo y
preguntas de la base de datos
       */
      public void pruebaExamenesConPreguntas(){

            AdministradorExamenes administrador = getAdministradorExamenes();

            ExamenDTO examen = new ExamenDTO();

            examen = administrador.leerExamen(8);

            System.out.println("Examen: " + examen.getTitulo());

            PreguntaDTO pregunta = new PreguntaDTO();

            pregunta.setPregunta("¿En que año se creo Java?");
            pregunta.setStatus(0);

            List<PreguntaDTO> lista = new ArrayList<PreguntaDTO>();
            lista.add(pregunta);

            examen.setPreguntas(lista);

            administrador.actualizarExamen(examen);

      }

      /**
       * metodo que prueba la lectura de todos los pacientes de la base de datos
       */
      public void pruebaLecturaPacientesTodos(){
```

```
            AdministradorPacientes administrador = getAdministradorPacientes();
            List<PacienteDTO> lista = new ArrayList<PacienteDTO>();
            lista=administrador.leerPacientes();
            System.out.println("Tamaño: " + lista.size());
      }

      /**
       * metodo que prueba la lectura de todos los medicos adscritos de la base de
datos
       */
      public void pruebaLecturaMedicosAdscritosTodos(){

            AdministradorMedicosAdscritos administrador =
getAdministradorMedicosAdscritos();
            List<MedicoAdscritoDTO> lista = new ArrayList<MedicoAdscritoDTO>();
            lista=administrador.leerMedicosAdscritos();
            System.out.println("Tamaño: " + lista.size());
      }

      /**
       * metodo que prueba la lectura de todos los administradores de la base de
datos
       */
      private void pruebaLecturaAdministradoresTodos(){

            AdministradorAdministradores administrador =
getAdministradorAdministradores();
            List<AdministradorDTO> lista = new ArrayList<AdministradorDTO>();
            lista=administrador.leerAdministradores();
            System.out.println("Tamaño: " + lista.size());
      }

      /**
       * metodo que prueba la lectura de todos los medicos residentes de la base
de datos
       */
      private void pruebaLecturaMedicosResidentesTodos(){

            AdministradorMedicosResidentes administrador =
getAdministradorMedicosResidentes();
            List<MedicoResidenteDTO> lista = new ArrayList<MedicoResidenteDTO>();
            lista = administrador.leerMedicosResidentes();
            System.out.println("No: " + lista.size());
      }

      /**
       * metodo que prueba la lectura y actualizacion de un paciente de la base de
datos
       */
      private void pruebaLecturayActualizacionPacientes(){

            AdministradorPacientes administrador = getAdministradorPacientes();
            PacienteDTO paciente = new PacienteDTO();
```

```
            paciente = administrador.leerPaciente(6);
            System.out.println("Nombres: " + paciente.getNombres());
            System.out.println("Enfermedades: " + paciente.getEnfermedades());
            paciente.setStatus(1);
            paciente.setEnfermedades("El paciente está sano");
            administrador.actualizacionPaciente(paciente);

      }

      /**
       * metodo que prueba la asignacion de un examen hacia un medico residente,
respetando la relacion muchos a muchos
       */
      private void pruebaAsignacionExamen(){

            AdministradorMedicosResidentes administrador =
getAdministradorMedicosResidentes();
            //AdministradorExamenes adminex = getAdministradorExamenes();

            MedicoResidenteDTO mr = new MedicoResidenteDTO();
            mr = administrador.leerMedicoResidenteId(7);
            System.out.println("Medico Residente:"  + mr.getNombres());
            ExamenDTO examen = new ExamenDTO();

            examen = administrador.leerExamenId(8);

            System.out.println("Examen: " + examen.getTitulo());

            mr.agregarExamen(examen);

            administrador.actualizarMedicoResidente(mr);
      }

      /**
       * metodo que prueba la insercion de un examen a la base de datos
       */
      public void pruebaInsertarExamen(){

            AdministradorExamenes administrador = getAdministradorExamenes();

            ExamenDTO examen = new ExamenDTO();

            examen.setTitulo("Java2EE");

            administrador.insertarExamen(examen);

      }

      /**
       * metodo que prueba la actualizacion y deshabilitacion de un medico
residente de la base de datos
       */
      public void pruebaEliminacionCatalogos(){
```

```
            AdministradorMedicosResidentes administrador =
getAdministradorMedicosResidentes();

            MedicoResidenteDTO mr = new MedicoResidenteDTO();
            mr = administrador.leerMedicoResidenteId(1);
            mr.setStatus(0);
            administrador.actualizarMedicoResidente(mr);
      }

      /**
       * metodo que prueba la lectura y actualizacion de un administrador de la
base de datos
       */
      public void pruebaLecturayActualizacionAdministrador(){

            AdministradorAdministradores administrador =
getAdministradorAdministradores();
            AdministradorDTO admin = new AdministradorDTO();
            admin = administrador.leerAdministrador(4);
            System.out.println("ID: " + admin.getId() + " | EMAIL ALTERNATIVO: "  +
admin.getEmail_alternativo());
            admin.setEmail_alternativo("elnuevo666@elnuevo.com");
            System.out.println("EMAIL ALTERNATIVO NUEVO: " +
admin.getEmail_alternativo());
            administrador.actualizarAdministrador(admin);
      }

      /**
       * metodo que prueba la lectura y actualizacion de un medico adscrito de la
base de datos
       */
      public void pruebaLecturayActualizacionMedicosAdscritos(){

            AdministradorMedicosAdscritos administrador =
getAdministradorMedicosAdscritos();

            MedicoAdscritoDTO mad = new MedicoAdscritoDTO();
            mad = administrador.leerMedicoAdscrito(2);
            System.out.println("Id: " + mad.getId() + " Area: " + mad.getArea() +
"Email:" + mad.getEmail());
            mad.setEmail("elmeronuevo@hotmail.com");
            mad.setArea("Nueva Area22");
            System.out.println("Email" + mad.getEmail());
            administrador.actualizarMedicoAdscrito(mad);

      }

      /**
       * metodo que prueba la lectura y actualizacion de un medico residente de la
base de datos
       */
      public void pruebaLecturayActualizacionMedicoResidente(){

            AdministradorMedicosResidentes administrador =
```

```
getAdministradorMedicosResidentes();

            MedicoResidenteDTO mr = new MedicoResidenteDTO();

            mr = administrador.leerMedicoResidenteId(7);

            System.out.println("ID: "+ mr.getId() + " status" + mr.getStatus());

            mr.setStatus(0);

            administrador.actualizarMedicoResidente(mr);
            System.out.println("ID: "+ mr.getId() + " status" + mr.getStatus());
      }

      /**
       * metodo que prueba la lectura de un paciente de la base de datos
       */
      public void pruebaLecturaPaciente(){

            AdministradorPacientes administrador = getAdministradorPacientes();

            PacienteDTO paciente = new PacienteDTO();

            paciente = administrador.leerPaciente(10);

            System.out.println("ID: " + paciente.getId() + " Nombres: " +
paciente.getNombres());

            paciente.setNombres("platano show");
            System.out.println("ID: " + paciente.getId() + " Nombres: " +
paciente.getNombres());

            administrador.actualizacionPaciente(paciente);
      }

      /**
       * metodo que prueba la lectura y actualizacion de un paciente
       */
      public void pruebaActualizacionPaciente(){


      }

      /**
       * metodo que prueba la insercion de un paciente a la base de datos
       * @throws ParseException
       */
      public void pruebaInsercionPaciente() throws ParseException{

            AdministradorPacientes administrador = getAdministradorPacientes();
            PacienteDTO pa = new PacienteDTO();

            pa.setAlergias("Alergias");
            pa.setAp_materno("MaternoPaciente");
```

```
            pa.setAp_paterno("PaternoPaciente");
            pa.setCelular("7777777777");
            pa.setCodigo_postal("75757");
            pa.setCountry_code("MEX");
            pa.setCuidados("Cuidados cuidados cuidados cuidados cuidados");
            pa.setDireccion("direccion paciente numero 12");
            pa.setEmail("paciente@hotmail.com");
            pa.setEnfermedades("enfermedades paciente enfermedades paciente
enfermedades paciente");

            SimpleDateFormat formatoFecha = new SimpleDateFormat( "dd-MM-yyyy" );
// dd/MM/yyyy
            Date fecha = null;

            try {
                  fecha = formatoFecha.parse( "03-01-1987" );
            } catch ( ParseException e ) {
                  e.printStackTrace();
                  System.out.println("Fecha inválida");
            }


            pa.setFecha_ingreso(fecha);
            pa.setFecha_nacimiento(fecha);

            pa.setGenero('M');
            pa.setLogin("juan");
            pa.setNo_afiliacion("654654654654654654");
            pa.setNombres("Nombre del Paciente");
            pa.setPassword("juan");
            pa.setPerfil(4);
            pa.setTel_casa("55332244");
            pa.setTipo_sangre("o positivo");

            administrador.insertarPaciente(pa);
      }

      /**
       * metodo que prueba la insercion de un administrador hacia la base de datos
       */
      public void pruebaInsercionAdministrador() {

            AdministradorAdministradores administrador =
getAdministradorAdministradores();

            AdministradorDTO ad = new AdministradorDTO();

            ad.setAp_materno("maternoAdmin");
            ad.setAp_paterno("paternoAdmin");
            ad.setCelular("23523523235");
            ad.setCodigo_postal("89797");
            ad.setCountry_code("MEX");
            ad.setDireccion("direccionAdmin");
            ad.setEmail("admin@administradores.com");
```

```
            ad.setEmail_alternativo("admin@hotmail.com");
            Date fecha = new Date();
            ad.setFecha_ingreso(fecha);
            ad.setFecha_nacimiento(fecha);
            ad.setGenero('H');
            ad.setLogin("admin");
            ad.setNombres("Nombre de Admin");
            ad.setPassword("admin");
            ad.setPerfil(3);
            ad.setTel_casa("55779988");

            administrador.insertarAdministrador(ad);

      }

      /**
       * metodo que prueba la insercion de un medico adscrito hacia la base de
datos
       */
      public void pruebaInsercionMEdicoAdscrito() {

            AdministradorMedicosAdscritos administrador =
getAdministradorMedicosAdscritos();

            MedicoAdscritoDTO ma = new MedicoAdscritoDTO();

            ma.setAp_materno("MaternoAdscrito");
            ma.setAp_paterno("Paternoadscrito");
            ma.setArea("Neoplasias");
            ma.setCelular("0445555889966");
            ma.setCodigo_postal("66699");
            ma.setCountry_code("MEX");
            ma.setDireccion("Hospital direccion medico adscrito");
            ma.setEmail("medico_adscrito@hospital.com");
            Date fecha = new Date();
            ma.setFecha_ingreso(fecha);
            ma.setFecha_nacimiento(fecha);
            ma.setGenero('H');
            ma.setLogin("adscrito");
            ma.setNombres("Medico Adscrito Nombre");
            ma.setPassword("adscrito");
            ma.setPerfil(2);
            ma.setPuesto("Jefe de Area");
            ma.setTel_casa("65454655");
            administrador.insertarMedicoAdscrito(ma);
      }

      /**
       * metodo que prueba la insercion de un medico residente respetando la
herencia de clases entre UsuarioDTO y MedicoResidenteDTO
       */
      public void pruebaHerencia() {

            AdministradorMedicosResidentes administrador =
```

```
getAdministradorMedicosResidentes();

            MedicoResidenteDTO mr = new MedicoResidenteDTO();

            mr.setAp_materno("Gopar");
            mr.setAp_paterno("Ramirez");
            mr.setCelular("5551736958");
            mr.setCodigo_postal("12244");
            mr.setCountry_code("MEX");
            mr.setDireccion("Tanque de gas");
            mr.setEmail("homero_sandy@hotmail.com");
            mr.setEscuela("UNAM");
            Date fecha = new Date();
            SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
            String cadenaFecha = formato.format(fecha);
            mr.setFecha_ingreso(fecha);
            mr.setFecha_nacimiento(fecha);
            mr.setGenero('M');
            mr.setLogin("sandy");
            mr.setNombres("Sandy Bel");
            mr.setPassword("sandy");
            mr.setPerfil(1);
            mr.setTel_casa("53045742");
            mr.setTel_trabajo("0000000");
            mr.setStatus(1);

            //ExamenDTO ex = new ExamenDTO();
            //ex.setTitulo("AASSDDFF");
            /*List <ExamenDTO> examenes = new ArrayList<ExamenDTO>();
            mr.setExamenes(examenes);*/
            //mr.agregarExamen(ex);
            administrador.insertarMedicoResidente(mr);
            //administrador.insertarMedicoResidente(mr);
      }

      /**
       * metodo que prueba la lectura de un medico residente de la base de datos
       */
      public void pruebaLectura(){

            AdministradorMedicosResidentes ad =
getAdministradorMedicosResidentes();

            MedicoResidenteDTO mr = new MedicoResidenteDTO();
            ExamenDTO ex = new ExamenDTO();
            mr = ad.leerMedicoResidenteId(1);
            ex = ad.leerExamenId(1);

            ad.asignarExamenes(mr, ex);

            System.out.println("Nombre: " + mr.getNombres());

      }
```

```
      private AdministradorMedicosResidentes getAdministradorMedicosResidentes()
        {
          ApplicationContext context = new ClassPathXmlApplicationContext
            ( "spring-services.xml" );

          return ( AdministradorMedicosResidentes)context.getBean(
"administradorMedicosResidentes" );
        }

      private AdministradorMedicosAdscritos getAdministradorMedicosAdscritos()
        {
          ApplicationContext context = new ClassPathXmlApplicationContext
            ( "spring-services.xml" );

          return ( AdministradorMedicosAdscritos)context.getBean(
"administradorMedicosAdscritos" );
        }

      private AdministradorAdministradores getAdministradorAdministradores()
        {
          ApplicationContext context = new ClassPathXmlApplicationContext
            ( "spring-services.xml" );

          return (AdministradorAdministradores)context.getBean(
"administradorAdministradores" );
        }

      private AdministradorPacientes getAdministradorPacientes()
        {
          ApplicationContext context = new ClassPathXmlApplicationContext
            ( "spring-services.xml" );

          return ( AdministradorPacientes)context.getBean(
"administradorPacientes" );
        }

      private AdministradorExamenes getAdministradorExamenes()
        {
          ApplicationContext context = new ClassPathXmlApplicationContext
            ( "spring-services.xml" );

          return (AdministradorExamenes)context.getBean(
"administradorExamenes" );
        }

      private AdministradorPreguntas getAdministradorPreguntas()
      {
            ApplicationContext context = new ClassPathXmlApplicationContext
                  ( "spring-services.xml" );
            return
(AdministradorPreguntas)context.getBean("administradorPreguntas");
      }

      private AdministradorCasos getAdministradorCasos()
```

```
	{
		ApplicationContext context = new ClassPathXmlApplicationContext
		("spring-services.xml");
		return (AdministradorCasos)context.getBean("administradorCasos");
	}

	private AdministradorContenidos getAdministradorContenidos()
	{
		ApplicationContext context = new ClassPathXmlApplicationContext
		("spring-services.xml");
		return
(AdministradorContenidos)context.getBean("administradorContenidos");
	}

}
```

8.5 paquete raíz

8.5.1 spring-dev.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd"
default-autowire="byName" default-lazy-init="true"
>
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
<property name="driverClassName">
<value>com.mysql.jdbc.Driver</value>
</property>
<property name="url">
<value>jdbc:mysql://localhost:3306/ptHospital?autoReconnect=true</value>
</property>
<property name="maxWait" value="10000"/>
<property name="username" value="root"/>
<property name="password" value="qazwsx"/>
</bean>
<bean id="transactionManager"
class="org.springframework.orm.hibernate3.HibernateTransactionManager" />
<tx:annotation-driven />
```

```
<bean id="sessionFactory"
class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"
>
<property name="annotatedClasses">
<list>
<value>mx.uam.azc.pt.data.UsuarioDTO</value>
<value>mx.uam.azc.pt.data.MedicoResidenteJoinDTO</value>
<value>mx.uam.azc.pt.data.MedicoResidenteSaveDTO</value>
<value>mx.uam.azc.pt.data.MedicoAdscritoJoinDTO</value>
<value>mx.uam.azc.pt.data.MedicoAdscritoSaveDTO</value>
<value>mx.uam.azc.pt.data.AdministradorJoinDTO</value>
<value>mx.uam.azc.pt.data.AdministradorSaveDTO</value>
<value>mx.uam.azc.pt.data.PacienteJoinDTO</value>
<value>mx.uam.azc.pt.data.PacienteSaveDTO</value>
<value>mx.uam.azc.pt.data.ExamenJoinDTO</value>
<value>mx.uam.azc.pt.data.ExamenSaveDTO</value>
<value>mx.uam.azc.pt.data.PreguntaDTO</value>
<value>mx.uam.azc.pt.data.RespuestaDTO</value>
<value>mx.uam.azc.pt.data.CasoJoinDTO</value>
<value>mx.uam.azc.pt.data.CasoSaveDTO</value>
<value>mx.uam.azc.pt.data.ContenidoJoinDTO</value>
<value>mx.uam.azc.pt.data.ContenidoSaveDTO</value>
<value>mx.uam.azc.pt.data.SubContenidoDTO</value>
<value>mx.uam.azc.pt.data.ReferenciaDTO</value>
<value>mx.uam.azc.pt.data.TemaRelacionadoJoinDTO</value>
<value>mx.uam.azc.pt.data.TemaRelacionadoSaveDTO</value>
</list>
</property>

<property name="mappingResources">
    <list><value>queries.hbm.xml</value></list>
</property>

<property name="hibernateProperties">
<props>
<prop key="hibernate.dialect">
org.hibernate.dialect.MySQLDialect
</prop>
</props>
</property>
</bean>
</beans>
```

8.5.2 spring-services

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://www.springframework.org/schema/beans
```

```
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd"
default-autowire="byName"
default-lazy-init="true"
>
<import resource="spring-dev.xml"/>

<bean id="administradorMedicosResidentes"
class="mx.uam.azc.pt.business.local.AdministradorMedicosResidentesImpl"></bean>
<bean id="administradorMedicosAdscritos"
class="mx.uam.azc.pt.business.local.AdministradorMedicosAdscritosImpl"></bean>
<bean id="administradorAdministradores"
class="mx.uam.azc.pt.business.local.AdministradorAdministradoresImpl"></bean>
<bean id="administradorPacientes"
class="mx.uam.azc.pt.business.local.AdministradorPacientesImpl"></bean>
<bean id="administradorExamenes"
class="mx.uam.azc.pt.business.local.AdministradorExamenesImpl"></bean>
<bean id="administradorPreguntas"
class="mx.uam.azc.pt.business.local.AdministradorPreguntasImpl"></bean>
<bean id="administradorCasos"
class="mx.uam.azc.pt.business.local.AdministradorCasosImpl"></bean>
<bean id="administradorContenidos"
class="mx.uam.azc.pt.business.local.AdministradorContenidosImpl"></bean>
<bean id="administradorUsuarios"
class="mx.uam.azc.pt.business.local.AdministradorUsuariosImpl"></bean>
</beans>
```

8.5.3 queries.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

      <query name="leer_medicos_residentes">
            FROM MedicoResidenteJoinDTO mr WHERE mr.status=1
      </query>

      <query name="leer_medicos_residentes_filtro">
            FROM MedicoResidenteJoinDTO mr WHERE (mr.nombres LIKE :nombres) AND
mr.status=0
      </query>

      <query name="leer_medicos_adscritos_filtro">
            FROM MedicoAdscritoJoinDTO ma WHERE (ma.nombres LIKE :nombres) AND
ma.status=0
```

```
	</query>

	<query name="leer_administradores_filtro">
		FROM AdministradorJoinDTO ad WHERE (ad.nombres LIKE :nombres) AND
ad.status=0
	</query>

	<query name="leer_pacientes_filtro">
		FROM PacienteJoinDTO p WHERE (p.nombres LIKE :nombres) AND p.status=0
	</query>

	<query name="leer_casos_filtro">
		FROM CasoJoinDTO ca WHERE (ca.titulo LIKE :titulo) AND ca.status=0
	</query>

	<query name="leer_contenidos_filtro">
		FROM ContenidoJoinDTO con WHERE (con.titulo_contenido LIKE
:titulo_contenido) AND con.status=0
	</query>

	<query name="leer_examenes_filtro">
		FROM ExamenJoinDTO ex WHERE (ex.titulo LIKE :titulo) AND ex.status=0
	</query>

	<query name="leer_administradores">
		FROM AdministradorJoinDTO ad
	</query>

	<query name="leer_medicos_adscritos">
		FROM MedicoAdscritoJoinDTO ma
	</query>

	<query name="leer_pacientes">
		FROM PacienteJoinDTO pa
	</query>

	<query name="leer_casos">
		FROM CasoJoinDTO ca
	</query>

	<query name="leer_contenidos">
		FROM ContenidoJoinDTO co
	</query>

	<query name="leer_subcontenidos">
		FROM SubContenidoDTO sc
	</query>

	<query name="leer_referencias">
		FROM ReferenciaDTO re
	</query>

	<query name="leer_temas_relacionados">
		FROM TemaRelacionadoJoinDTO tr
```

```
	</query>

	<query name="leer_examenes">
		FROM ExamenJoinDTO ex
	</query>

	<query name="leer_preguntas">
		FROM PreguntaDTO pr
	</query>

	<query name="leer_respuestas">
		FROM RespuestaDTO res
	</query>

	<query name="leer_usuarios">
		FROM UsuarioDTO usu
	</query>

	<query name="leer_usuario_login_pass">
		FROM UsuarioDTO u WHERE
		(
		(u.login = ? )AND
		(u.password = ?)
		)
	</query>


</hibernate-mapping>
```

8.5.4 base.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
	<package name="base" extends="struts-default" abstract="true">

		<interceptors>

			<interceptor name="administradores_interceptor"

	class="mx.uam.azc.pt.interceptors.AdministradoresInterceptor"></interceptor>

			<interceptor name="pacientes_interceptor"

	class="mx.uam.azc.pt.interceptors.PacientesInterceptor"></interceptor>

			<interceptor name="medicos_residentes_interceptor"
```

```
class="mx.uam.azc.pt.interceptors.MedicosResidentesInterceptor"></interceptor>

                    <interceptor name="map_to_request"

      class="com.acsinet_solutions.interceptors.MapToRequestInterceptor" />

                  <interceptor name="bean_scope"
                        class="com.googlecode.scopeplugin.ScopeInterceptor" />

                  <interceptor-stack name="customStack">
                        <interceptor-ref name="exception" />
                        <interceptor-ref name="map_to_request" />
                        <interceptor-ref name="servletConfig" />
                        <interceptor-ref name="bean_scope" />
                        <interceptor-ref name="prepare" />
                        <interceptor-ref name="i18n" />
                        <interceptor-ref name="debugging" />
                        <interceptor-ref name="profiling" />
                        <interceptor-ref name="checkbox" />
                        <interceptor-ref name="params" />
                        <interceptor-ref name="conversionError" />
                        <interceptor-ref name="validation">
                              <param
name="excludeMethods">insertar,actualizar,cancel,browse</param>
                        </interceptor-ref>
                        <interceptor-ref name="workflow">
                              <param
name="excludeMethods">insertar,actualizar,cancel,browse</param>
                        </interceptor-ref>
                  </interceptor-stack>
                  <interceptor-stack name="administradorStack">
                        <interceptor-ref name="administradores_interceptor" />
                        <interceptor-ref name="customStack" />
                  </interceptor-stack>

                  <interceptor-stack name="pacienteStack">
                        <interceptor-ref name="pacientes_interceptor" />
                        <interceptor-ref name="customStack" />
                  </interceptor-stack>

                  <interceptor-stack name="medicosResidenteStack">
                        <interceptor-ref name="medicos_residentes_interceptor" />
                        <interceptor-ref name="customStack" />
                  </interceptor-stack>

            </interceptors>

            <default-interceptor-ref name="customStack" />

      </package>
</struts>
```

8.5.5 struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

      <constant name="struts.devMode" value="true" />
      <constant name="struts.objectFactory"
            value="org.apache.struts2.spring.StrutsSpringObjectFactory" />
      <constant name="struts.objectFactory.spring.autoWire" value="name" />
      <constant name="struts.ui.theme" value="simple" />

      <include file="base.xml"></include>
      <include file="struts-extensions.xml" />

      <package name="pt" extends="base">

            <!-- <global-results> <result name="error_page">/WEB-
INF/pages/error.jsp</result>
                  <result name="sql_error_page">/WEB-
INF/pages/sql_error.jsp</result> </global-results>
                  <global-exception-mappings> <exception-mapping
exception="java.sql.SQLException"
                  result="sql_error_page" /> <exception-mapping
exception="java.lang.Exception"
                  result="error_page" /> </global-exception-mappings> -->

            <action name="principal" class="mx.uam.azc.pt.actions.PrincipalAction">
                  <result>/WEB-INF/principal.jsp</result>
            </action>

            <action name="administradores_*"
class="mx.uam.azc.pt.actions.AdministradorAction"
                  method="{1}">
                  <interceptor-ref name="administradorStack" />
                  <result name="menu">/WEB-
INF/pages/administradores/menu.jsp</result>
                  <result name="listar_medicos_residentes">/WEB-
INF/pages/administradores/listar_medicos_residentes.jsp
                  </result>
                  <result name="ver_medicos_residentes">/WEB-
INF/pages/administradores/ver_medico_residente.jsp
                  </result>
                  <result name="forma_insertar_medicos_residentes">/WEB-
INF/pages/administradores/forma_insertar_medicos_residentes.jsp
                  </result>
                  <result name="forma_actualizar_medicos_residentes">/WEB-
INF/pages/administradores/forma_actualizar_medicos_residentes.jsp
                  </result>
                  <result name="listar_examenes">/WEB-
```

```
INF/pages/administradores/listar_examenes.jsp
                </result>
                <result name="listar_medicos_residentes_redirect"
type="redirectAction">administradores_listar_medicos_residentes
                </result>
                <result name="ver_medicos_residentes_redirect"
type="redirectAction">medicos_residentes_ver
                </result>
                <result name="error" type="redirectAction">
                      <param name="actionName">login_forma_login</param>
                </result>


                <result name="listar_medicos_adscritos">/WEB-
INF/pages/administradores/listar_medicos_adscritos.jsp
                </result>
                <result name="ver_medicos_adscritos">/WEB-
INF/pages/administradores/ver_medico_adscrito.jsp
                </result>
                <result name="forma_insertar_medicos_adscritos">/WEB-
INF/pages/administradores/forma_insertar_medicos_adscritos.jsp
                </result>
                <result name="forma_actualizar_medicos_adscritos">/WEB-
INF/pages/administradores/forma_actualizar_medicos_adscritos.jsp
                </result>
                <result name="listar_medicos_adscritos_redirect"
type="redirectAction">administradores_listar_medicos_adscritos
                </result>
                <result name="ver_medicos_adscritos_redirect"
type="redirectAction">medicos_adscritos_ver
                </result>
                <!-- <result name="error" type="redirectAction"> <param
name="actionName">login_forma_login</param>
                      </result> -->


                <result name="listar_administradores">/WEB-
INF/pages/administradores/listar_administradores.jsp
                </result>
                <result name="ver_administradores">/WEB-
INF/pages/administradores/ver_administrador.jsp
                </result>
                <result name="forma_insertar_administradores">/WEB-
INF/pages/administradores/forma_insertar_administradores.jsp
                </result>
                <result name="forma_actualizar_administradores">/WEB-
INF/pages/administradores/forma_actualizar_administradores.jsp
                </result>
                <result name="listar_administradores_redirect"
type="redirectAction">administradores_listar_administradores
                </result>
                <result name="ver_administradores_redirect"
type="redirectAction">administradores_ver
                </result>
```

```
                <!-- <result name="error" type="redirectAction"> <param
name="actionName">login_forma_login</param>
                      </result> -->


                <result name="listar_pacientes">/WEB-
INF/pages/administradores/listar_pacientes.jsp
                </result>
                <result name="ver_pacientes">/WEB-
INF/pages/administradores/ver_paciente.jsp
                </result>
                <result name="forma_insertar_pacientes">/WEB-
INF/pages/administradores/forma_insertar_pacientes.jsp
                </result>
                <result name="forma_actualizar_pacientes">/WEB-
INF/pages/administradores/forma_actualizar_pacientes.jsp
                </result>
                <result name="listar_pacientes_redirect"
type="redirectAction">administradores_listar_pacientes
                </result>
                <result name="ver_pacientes_redirect"
type="redirectAction">pacientes_ver
                </result>
                <!-- <result name="error" type="redirectAction"> <param
name="actionName">login_forma_login</param>
                      </result> -->

                <result name="listar_casos">/WEB-
INF/pages/administradores/listar_casos.jsp
                </result>
                <result name="ver_casos">/WEB-
INF/pages/administradores/ver_caso.jsp
                </result>
                <result name="forma_insertar_casos">/WEB-
INF/pages/administradores/forma_insertar_casos.jsp
                </result>
                <result name="forma_actualizar_casos">/WEB-
INF/pages/administradores/forma_actualizar_casos.jsp
                </result>
                <result name="listar_casos_redirect"
type="redirectAction">administradores_listar_casos
                </result>
                <result name="ver_casos_redirect" type="redirectAction">casos_ver
                </result>

                <result name="listar_contenidos">/WEB-
INF/pages/administradores/listar_contenidos.jsp
                </result>
                <result name="ver_contenidos">/WEB-
INF/pages/administradores/ver_contenido.jsp
                </result>
                <result name="forma_insertar_contenidos">/WEB-
INF/pages/administradores/forma_insertar_contenidos.jsp
                </result>
```

```
                <result name="forma_actualizar_contenidos">/WEB-
INF/pages/administradores/forma_actualizar_contenidos.jsp
                </result>
                <result name="listar_contenidos_redirect"
type="redirectAction">administradores_listar_contenidos
                </result>
                <result name="ver_contenidos_redirect"
type="redirectAction">contenidos_ver
                </result>
                <!-- <result
name="listar_temas_relacionados_redirect">listar_temas_relacionados
                </result> -->
                <result name="listar_temas_relacionados">/WEB-
INF/pages/administradores/listar_temas_relacionados.jsp
                </result>
                <result name="listar_remas_relacionados_redirect"
type="redirectAction">administradores_listar_temas_relacionados
                </result>

                <result name="listar_examenes">/WEB-
INF/pages/administradores/listar_examenes.jsp
                </result>
                <result name="ver_examenes">/WEB-
INF/pages/administradores/ver_examen.jsp
                </result>
                <result name="forma_insertar_examenes">/WEB-
INF/pages/administradores/forma_insertar_examenes.jsp
                </result>
                <result name="forma_actualizar_examenes">/WEB-
INF/pages/administradores/forma_actualizar_examenes.jsp
                </result>
                <result name="listar_examenes_redirect"
type="redirectAction">administradores_listar_examenes
                </result>
                <result name="ver_examenes_redirect"
type="redirectAction">examenes_ver
                </result>

            </action>

            <action name="pacientes_*"
class="mx.uam.azc.pt.actions.AdministradorAction"
                method="{1}">
                <interceptor-ref name="pacienteStack" />
                <result name="listar_casos">/WEB-
INF/pages/pacientes/listar_casos.jsp
                </result>
                <result name="ver_casos">/WEB-INF/pages/pacientes/ver_caso.jsp
                </result>
                <result name="listar_casos_redirect"
type="redirectAction">pacientes_listar_casos
                </result>

            </action>
```

```
            <action name="medicos_residentes_*"
class="mx.uam.azc.pt.actions.AdministradorAction"
                  method="{1}">
                  <interceptor-ref name="medicosResidenteStack" />
                  <result name="listar_casos">/WEB-
INF/pages/medicos_residentes/listar_casos.jsp
                  </result>
                  <result name="ver_casos">/WEB-
INF/pages/medicos_residentes/ver_caso.jsp
                  </result>
                  <result name="listar_casos_redirect"
type="redirectAction">medicos_residentes_listar_casos
                  </result>

                  <result name="listar_contenidos">/WEB-
INF/pages/medicos_residentes/listar_contenidos.jsp
                  </result>
                  <result name="ver_contenidos">/WEB-
INF/pages/medicos_residentes/ver_contenido.jsp
                  </result>
                  <result name="listar_contenidos_redirect"
type="redirectAction">medicos_residentes_listar_contenidos
                  </result>

                  <result name="listar_examenes">/WEB-
INF/pages/medicos_residentes/listar_examenes.jsp
                  </result>
                  <result name="ver_examenes">/WEB-
INF/pages/medicos_residentes/ver_examen.jsp
                  </result>
                  <result name="listar_examenes_redirect"
type="redirectAction">medicos_residentes_listar_examenes
                  </result>

            </action>

            <action name="login_*" class="mx.uam.azc.pt.actions.LoginAction"
                  method="{1}">
                  <result name="menu">/WEB-
INF/pages/administradores/menu.jsp</result>
                  <result name="menu_pacientes">/WEB-
INF/pages/pacientes/menu.jsp</result>
                  <result name="menu_medicos_residentes">/WEB-
INF/pages/medicos_residentes/menu.jsp</result>
                  <result name="forma_login">/WEB-
INF/pages/login/login.jsp</result>
                  <result name="login_redirect"
type="redirectAction">login_login</result>
                  <result name="menuRedirect" type="redirectAction">
                        <param name="actionName">login_menu</param>
                  </result>
                  <result name="error" type="redirectAction">
                        <param name="actionName">login_forma_login</param>
```

```
                </result>

            </action>

      </package>
</struts>
```

8.6 mx.uam.azc.pt.interceptors

8.6.1 AdministradoresInterceptor

```
package mx.uam.azc.pt.interceptors;

import mx.uam.azc.pt.data.UsuarioDTO;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;
import java.util.Map;

/**
 * Clase que servira para el manejo de sesion de administradors y medicos
adscritoss
 * @author Roberto Gonzalez
 *
 */
public class AdministradoresInterceptor implements Interceptor {

      public void destroy() {
            System.out.println("Destruyendo TemasInterceptor...");
      }

      public void init() {
            System.out.println("Inicializando TemasInterceptor...");
      }

      /**
       * Metodo interceptor que validara que se cumpla el perfil del usuario
       */
      public String intercept(ActionInvocation invocation) throws Exception {
            Map<?, ?> session = invocation.getInvocationContext().getSession();
            UsuarioDTO usuario = (UsuarioDTO) session.get("usuario");
            if (usuario != null && (usuario.getPerfil() == 1 || usuario.getPerfil()
== 2))
                  return invocation.invoke();
            else
                  return "error";
      }
}
```

8.6.2 MedicosResidentesInterceptor

```
package mx.uam.azc.pt.interceptors;

import mx.uam.azc.pt.data.UsuarioDTO;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;
import java.util.Map;

/**
 * Clase que servira para el manejo de sesion de medicos residentes
 * @author Roberto Gonzalez
 *
 */
public class MedicosResidentesInterceptor implements Interceptor {

	public void destroy() {
		System.out.println("Destruyendo TemasInterceptor...");
	}

	public void init() {
		System.out.println("Inicializando TemasInterceptor...");
	}

	/**
	 * Metodo interceptor que validara que se cumpla el perfil del usuario
	 */
	public String intercept(ActionInvocation invocation) throws Exception {
		Map<?, ?> session = invocation.getInvocationContext().getSession();
		UsuarioDTO usuario = (UsuarioDTO) session.get("usuario");
		if (usuario != null && (usuario.getPerfil() == 3))
			return invocation.invoke();
		else
			return "error";
	}
}
```

8.6.3 PacientesInterceptor

```
package mx.uam.azc.pt.interceptors;

import mx.uam.azc.pt.data.UsuarioDTO;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;
import java.util.Map;

/**
 * Clase que servira para el manejo de sesion de pacientes
 * @author Roberto Gonzalez
 *
 */
```

```
public class PacientesInterceptor implements Interceptor {

	public void destroy() {
		System.out.println("Destruyendo TemasInterceptor...");
	}

	public void init() {
		System.out.println("Inicializando TemasInterceptor...");
	}

	/**
	 * Metodo interceptor que validara que se cumpla el perfil del usuario
	 */
	public String intercept(ActionInvocation invocation) throws Exception {
		Map<?, ?> session = invocation.getInvocationContext().getSession();
		UsuarioDTO usuario = (UsuarioDTO) session.get("usuario");
		if (usuario != null && (usuario.getPerfil() == 4))
			return invocation.invoke();
		else
			return "error";
	}
}
```

8.7 /css

8.7.1 datepicker.css

```
/* This is a very basic stylesheet for the date-picker. Feel free to create your
own. */

/* Hide the input by using a className */
input.fd-hidden-input,
select.fd-hidden-input
	{
	display:none;
	}
/* Screen reader class - hides it from the visual display */
.fd-screen-reader
	{
	position:absolute;
	left:-999em;
	top:0;
	width:1px;
	height:1px;
	overflow:hidden;
	outline: 0 none;
       -moz-outline: 0 none;
	}
```

```
/* Disabled datePicker and activation button */
a.dp-disabled,
.dp-disabled table
        {
        opacity:.3 !important;
        filter:alpha(opacity=40);
        }
.dp-disabled,
.dp-disabled td,
.dp-disabled th,
.dp-disabled th span
        {
        cursor:default !important;
        }
a.date-picker-control:focus,
div.datePicker table td:focus
        {
        overflow:hidden;
        outline:0 none;
      -moz-outline: 0 none;
      color:rgb(100,130,170) !important;
        }
/* The wrapper div */
div.datePicker
        {
        position:absolute;
        z-index:9999;
        text-align:center;

        /* Change the font-size to suit your design's CSS. The following line is
for the demo that has a 12px font-size defined on the body tag */
        font:900 0.8em/1em Verdana, Sans-Serif;

        /* For Example: If using the YUI font CSS, uncomment the following line to
get a 10px font-size within the datePicker */
        /* font:900 77%/77% Verdana, sans-serif; */

        /* Or, if you prefer a pixel precision */
        /* font:900 12px/12px Verdana, sans-serif; */

        background:transparent;

        /* Mozilla & Webkit extensions to stop text-selection. */
        -moz-user-select:none;
        -khtml-user-select:none;
        }
/* Styles for the static datePickers */
div.static-datepicker
        {
        position:relative;
        top:5px;
        left:0;
        }
div.datePicker table
```

```
        {
        width:auto;
        height:auto;
        }
/* Draggable datepickers */
div.datePicker tfoot th.drag-enabled,
div.datePicker thead th.drag-enabled,
div.datePicker thead th.drag-enabled span
        {
        cursor:move;
        }
/* The iframe hack to cover selectlists in Internet Explorer <= v6 */
iframe.iehack
        {
        position:absolute;
        background:#fff;
        z-index:9998;
        padding:0;
        border:0;
        display:none;
        margin:0;
        }
/* The "button" created beside each input for non-static datePickers */
a.date-picker-control:link,
a.date-picker-control:visited
        {
        position:relative;
        /* Moz & FF */
        display: -moz-inline-stack;
        border:0 none;
        padding:0;
        margin:0 0 0 4px;
        background:transparent url(../img/cal-grey.gif) no-repeat 50% 50%;
        min-width:16px;
        line-height:1;
        cursor:pointer;
        visibility:visible;
        text-decoration:none;
        vertical-align:top;
        }
a.date-picker-control:hover,
a.date-picker-control:active,
a.date-picker-control:focus,
a.dp-button-active:link,
a.dp-button-active:visited,
a.dp-button-active:hover,
a.dp-button-active:active,
a.dp-button-active:focus
        {
        background:transparent url(../img/cal.gif) no-repeat 50% 50% !important;
        }
/* Feed IE6 the following rule, IE7 should handle the min-width declared above */
* html a.date-picker-control
        {
```

```
        width:16px;
        }
/* IE, Safari & Opera. Seperate CSS rule seems to be required. */
a.date-picker-control
        {
        display:inline-block;
        }
a.date-picker-control span
        {
        display:block;
        width:16px;
        height:16px;
        margin:auto 0;
        }
/* Default "button" styles */
div.datePicker thead th span
        {
        display:block;
        padding:0;
        margin:0;
        text-align:center;
        line-height:1em;
        border:0 none;
        background:transparent;
        font-weight:bold;
        cursor:pointer;
        }
/* The "month, year" display */
div.datePicker th span.month-display,
div.datePicker th span.year-display
        {
        display:inline;
        text-transform:uppercase;
        letter-spacing:1px;
        font:normal 1.2em Verdana, Sans-Serif;
        cursor:default;
        }
/* Next & Previous (month, year) buttons */
div.datePicker th span.prev-but,
div.datePicker th span.next-but
        {
        font-weight:lighter;
        font-size:2.4em;
        font-family: georgia, times new roman, palatino, times, bookman, serif;
        cursor:pointer !important;
        }
/* Hover effect for Next & Previous (month, year) buttons */
div.datePicker th span.prev-but:hover,
div.datePicker th span.next-but:hover,
div.datePicker th span.today-but:hover
        {
        color:#a84444;
        }
/* Today button */
```

```
div.datePicker th span.today-but
        {
        text-align:center;
        margin:0 auto;
        font:normal 1em Verdana, Sans-Serif;
        width:100%;
        text-decoration:none;
        padding-top:0.3em;
        text-transform:uppercase;
        vertical-align:middle;
        cursor:pointer !important
        }
/* Disabled buttons */
div.dp-disabled th span.prev-but,
div.dp-disabled th span.next-but,
div.dp-disabled th span.today-but,
div.dp-disabled th span.prev-but:hover,
div.dp-disabled th span.next-but:hover,
div.dp-disabled th span.today-but:hover,
div.datePicker th span.prev-but.fd-disabled:hover,
div.datePicker th span.next-but.fd-disabled:hover,
div.datePicker thead th span.fd-disabled,
div.datePicker th span.fd-disabled:hover
        {
        color:#aaa;
        cursor:default !important;
        }
/* The mon, tue, wed etc day buttons */
div.datePicker th span.fd-day-header
        {
        text-align:center;
        margin:0 auto;
        font:900 1em Verdana, Sans-Serif;
        text-decoration:none;
        text-transform:lowercase;
        cursor:pointer;
        }
/* The table */
div.datePicker table
        {
        margin:0;
        padding:0px;
        border:1px solid #ccc;
        background:#fff url(../img/gradient-e5e5e5-ffffff.gif) repeat-x 0 -20px;
        text-align:center;
        border-spacing:2px;
        padding:0.3em;
        width:auto;
        empty-cells:show;
        -moz-border-radius:0.8em;
        border-radius:0.8em;
        font:normal 1em Verdana, Sans-Serif;
        }
/* Common TD & TH styling */
```

```
div.datePicker table td,
div.datePicker table tbody th
        {
        border:0 none;
        padding:0;
        text-align:center;
        vertical-align:middle;
        cursor:pointer;
        background:#fff url(../img/gradient-e5e5e5-ffffff.gif) repeat-x 0 -40px;
        width:1.5em;
        height:1.5em;
        overflow:hidden;
        outline:transparent none 0px;
        border:1px solid #ccc;
        text-transform:none;
        -moz-border-radius:2px;
        border-radius:2px;
        }
div.datePicker table td:focus,
div.datePicker table td:active
        {
        outline:0 none red;
        }
div.datePicker table th
        {
        border:0 none;
        padding:0;
        font-weight:bold;
        color:#222;
        text-align:center;
        vertical-align:middle;
        text-transform:none;
        }
div.datePicker table thead th
        {
        height:auto !important;
        }
div.datePicker table tbody th
        {
        border:1px solid #dcdcdc;
        }
/* Week number display */
div.datePicker table thead th.date-picker-week-header,
div.datePicker table tbody th.date-picker-week-header
        {
        font-style:oblique;
        background:transparent;
        cursor:default;
        }
div.datePicker table thead th.date-picker-week-header
        {
        cursor:help;
        border:0 none;
        padding:0 0 0.2em 0;
```

```
        }
/* tfoot status bar */
div.datePicker tfoot th
        {
        cursor:default;
        font-weight:normal;
        text-transform:uppercase;
        letter-spacing:0.1em;
        border:0 none;
        background:#fff;
        height:2.8em;
        }
/* TD cell that is _not_ used to display a day of the month */
div.datePicker table tbody td.date-picker-unused
        {
        background:#fff url(../img/backstripes.gif);
        border-color:#dcdcdc;
        cursor:default !important;
        }

/* The TH cell used to display the "month, year" title */
div.datePicker table thead th.date-picker-title
        {
        width:auto;
        height:auto;
        padding:0.4em 0;
        }
/* The "mon tue wed etc" day header styles */
div.datePicker table thead th.date-picker-day-header
        {
        text-transform:lowercase;
        cursor:help;
        height:auto;
        }
/* The "todays date" style */
div.datePicker table tbody td.date-picker-today
        {
        background:#fff url(../img/bullet2.gif) no-repeat 0 0;
        color:rgb(100,100,100) !important;
        }

div.datePicker table tbody td.month-out.date-picker-highlight
        {
        color:#aa8866 !important;
        }
/* The "highlight days" style */
div.datePicker table tbody td.date-picker-highlight,
div.datePicker table thead th.date-picker-highlight
        {
        color:#a86666 !important;
        }
/* The "active cursor" style */
div.datePicker table tbody td.date-picker-hover
        {
```

```
        background:#fff url(../img/bg_header.jpg) no-repeat 0 0;
        cursor:pointer;
        border-color:rgb(100,130,170) !important;
        color:rgb(100,130,170);
        text-shadow: 0px 1px 1px #fff;
        }
/* The "disabled days" style */
div.datePicker table tbody td.day-disabled
        {
        background:#fff url(../img/backstripes.gif) no-repeat 0 0;
        color:#aaa !important;
        cursor:default;
        text-decoration:line-through;
        }
div.datePicker table tbody td.month-out
        {
        border-color:#ddd;
        color:#aaa !important;
        background:#fff url(../img/gradient-e5e5e5-ffffff.gif) repeat-x 0 -40px;
        }
/* The "selected date" style */
div.datePicker table tbody td.date-picker-selected-date
        {
        color:#333 !important;
        border-color:#333 !important;
        }
/* The date "out of range" style */
div.datePicker table tbody td.out-of-range,
div.datePicker table tbody td.not-selectable
        {
        color:#ccc !important;
        font-style:oblique;
        background:#fcfcfc !important;
        cursor:default !important;
        }
/* Week number "out of range" && "month-out" styles */
div.datePicker table tbody th.month-out,
div.datePicker table tbody th.out-of-range
        {
        color:#aaa !important;
        font-style:oblique;
        background:#fcfcfc !important;
        }
/* week numbers "out of range" */
div.datePicker table tbody th.out-of-range
        {
        opacity:0.6;
        filter:alpha(opacity=60);
        }
/* Used when the entire grid is full but the next/prev months dates cannot be
selected */
div.datePicker table tbody td.not-selectable
        {
        opacity:0.8;
```

```
        filter:alpha(opacity=80);
        }
div.datePicker table tbody tr
        {
        display:table-row;
        }
div.datePicker table tfoot sup
        {
        font-size:0.86em;
        letter-spacing:normal;
        text-transform:none;
        height: 0;
      line-height: 1;
      position: relative;
      top: -0.2em;
      vertical-align: baseline !important;
      vertical-align: top;
        }
div.datePicker table thead th.date-picker-day-header,
div.datePicker table thead span.month-display,
div.datePicker table thead span.year-display
        {
        text-shadow: 0px 1px 1px #fff;
        }
/* You can add focus effects (for everything but IE6) like so: */
div.datepicker-focus
        {
        /* Naughty, naughty - but we add a highlight using the table's border
colour */
        outline:none;
        }
div.datepicker-focus table.datePickerTable
        {
        border-color:#999 !important;
        }
div.datePicker table tbody tr td:focus
        {
        overflow:hidden;
        outline:0 none;
      -moz-outline: 0 none;
      color:rgb(100,130,170) !important;
        }
/* INTERNET EXPLORER WOES
   ======================

   Hover Effects
   -------------

   IE cannot deal with :focus on the TR so the datePicker script adds the class
"dp-row-highlight" to the
   row currently being hovered over. This should enable you to add hover effects
if desired.

   e.g. the following rule will highlight the cell borders in another colour when
```

```
a row is moused over,
   it looks like crap though so I didn't include the rule within the demo:

div.datePicker table tbody tr.dp-row-highlight td
        {
        border-color:#aaa;
        }
*/

/* Remove the images for Internet Explorer <= v6 using the "* html" hack
   This is a workaround for a nasty IE6 bug that never caches background images on
dynamically created DOM nodes
   which means that they are downloaded for every cell for every table - nasty! */
* html div.datePicker table td
        {
        background-image:none;
        }
* html div.datePicker table td.date-picker-unused
        {
        background:#f2f2f2;
        }
/* Chrome has problems with the -webkit-box-shadow and -webkit-border-radius
styles together
   Remove one or the other to get things looking less ugly */
@media screen and (-webkit-min-device-pixel-ratio:0) {
        div.datePicker table
                {
                border-spacing:0.3em;
                /* Naughty, naughty */
                -webkit-box-shadow:0px 0px 5px #aaa;
                -webkit-border-radius:0.8em;
                }
        div.static-datepicker table
                {
                -webkit-box-shadow:0 0 0 transparent;
                }
        div.static-datepicker:focus table
                {
                -webkit-box-shadow:0px 0px 5px #aaa;
                }
        div.datePicker table td,
        div.datePicker table tbody th
                {
                padding:0.1em;
                -webkit-border-radius:2px;
                }
        div.datePicker table tbody td.date-picker-hover
                {
                -webkit-box-shadow:0px 0px 1px rgb(100,130,170);
                }
}
/* Untested webkit rules for fading out the disabled buttons - fingers crossed */
@-webkit-keyframes fadeout {
        to {
```

```
                opacity: 0.4;
        }
        from {
                opacity: 1.0;
                color:#222;
        }
}
@media screen and (-webkit-min-device-pixel-ratio:0) {
        div.datePicker table thead th span.fd-disabled {
                -webkit-animation-name: fadeout;
                -webkit-animation-duration: 3s;
                -webkit-animation-timing-function: ease-in-out;
        }
}
```

8.7.2 style.css

```
body {
      background-color: #FFFFFF;
      font-family: Helvetica, sans-serif;
}

:link {
      color: #0000FF;
}

:vlink {
      color: #0000FF;
}

:alink {
      color: #FF3300;
}

.important {
      font-weight: bold;
      color: red;
      display: inline;
      font-size: smaller;
}

.section {
      font-weight: bold;
      color: #009999;
      margin: 10 10 5 5;
}

.step {
      font-weight: bold;
      color: #FF0000;
      margin: 5;
```

```
}

.instructions {
	font-size: smaller;
}

.instructionlist {
	text-align: left;
}

.form {
	background-color: #000099;
}

.label {
	font-size: smaller;
	font-weight: bold;
	color: white;
	margin: 5 5 5 5;
}

.item {
	font-weight: bold;
	margin: 0 5 30 5;
	text-align: left;
}

.smallitem {
	font-size: smaller;
}

////////////////////////////
Live Validation ////////////////////////////////
.LV_validation_message {
	font-weight: bold;
	margin: 0 0 0 5px;
}

.LV_valid {
	color: #00CC00;
}

.LV_invalid {
	font-size: 10px;
	color: #CC0000;
}

.
LV_valid_field,input.LV_valid_field:hover,input.LV_valid_field:active,textarea.LV_
valid_field:hover,textarea.LV_valid_field:active
	{

}
```

```
.
LV_invalid_field,input.LV_invalid_field:hover,input.LV_invalid_field:active,textar
ea.LV_invalid_field:hover,textarea.LV_invalid_field:active
	{
	border: 2px solid #CC0000;
}

div.menuBar,div.menuBar a.menuButton,div.menu,div.menu a.menuItem {
	font-family: "MS Sans Serif", Arial, sans-serif;
	font-size: 8pt;
	font-style: normal;
	font-weight: normal;
	color: #000000;
}

div.menuBar {
	background-color: #CCE57F;
	border: 2px solid;
	border-color: #ECf59F #79aC00 #79aC00 #ECf59F;
	padding: 4px 2px 4px 2px;
	text-align: left;
}

div.menuBar a.menuButton {
	background-color: transparent;
	border: 1px solid #CCE57F;
	color: #000000;
	cursor: default;
	left: 0px;
	margin: 1px;
	padding: 2px 6px 2px 6px;
	position: relative;
	text-decoration: none;
	top: 0px;
	z-index: 100;
}

div.menuBar a.menuButton:hover {
	background-color: transparent;
	border-color: #ECf59F #79aC00 #79aC00 #ECf59F;
	color: #000000;
}

div.menuBar a.menuButtonActive,div.menuBar a.menuButtonActive:hover {
	background-color: #99CC00;
	border-color: #79aC00 #ECf59F #ECf59F #79aC00;
	color: #ffffff;
	left: 1px;
	top: 1px;
}

div.menu {
	background-color: #CCE57F;
	border: 2px solid;
```

```
	border-color: #ECf59F #79aC00 #79aC00 #ECf59F;
	left: 0px;
	padding: 0px 1px 1px 0px;
	position: absolute;
	top: 0px;
	visibility: hidden;
	z-index: 101;
}

div.menu a.menuItem {
	color: #000000;
	cursor: default;
	display: block;
	padding: 3px 1em;
	text-decoration: none;
	white-space: nowrap;
}

div.menu a.menuItem:hover,div.menu a.menuItemHighlight {
	background-color: #79aC00;
	color: #ffffff;
}

div.menu a.menuItem span.menuItemText {

}

div.menu a.menuItem span.menuItemArrow {
	margin-right: -.75em;
}

div.menu div.menuItemSep {
	border-top: 1px solid #79aC00;
	border-bottom: 1px solid #ECf59F;
	margin: 4px 2px;
}
```

8.8 /js

8.8.1 datepicker.js

```
/*
	DatePicker v4.5 by frequency-decoder.com

	Released under a creative commons Attribution-ShareAlike 2.5 license
(http://creativecommons.org/licenses/by-sa/2.5/)

	Please credit frequency-decoder in any derivative work - thanks.
```

```
        You are free:

        * to copy, distribute, display, and perform the work
        * to make derivative works
        * to make commercial use of the work

        Under the following conditions:

                by Attribution.
                --------------
                You must attribute the work in the manner specified by the author
or licensor.

                sa
                --
                Share Alike. If you alter, transform, or build upon this work, you
may distribute the resulting work only under a license identical to this one.

        * For any reuse or distribution, you must make clear to others the license
terms of this work.
        * Any of these conditions can be waived if you get permission from the
copyright holder.
*/

var datePickerController = (function datePickerController() {
        var languageInfo        = parseNavigatorLanguage(),
            datePickers         = {},
            uniqueId            = 0,
            weeksInYearCache    = {},
            localeImport        = false,
            nbsp                = String.fromCharCode(160),
            nodrag              = false,
            buttonTabIndex      = true,
            returnLocaleDate    = false,
            splitAppend         = ["-dd","-mm"],
            cellFormat          = "d-sp-F-sp-Y",
            titleFormat         = "F-sp-d-cc-sp-Y",
            formatParts         = ["placeholder", "sp-F-sp-Y"],
            formatMasks         = ["Y-sl-m-sl-d","m-sl-d-sl-Y","d-sl-m-sl-Y","Y-
ds-m-ds-d","m-ds-d-ds-Y","d-ds-m-ds-Y","d-ds-m-ds-y","d-sl-m-sl-y"],
            localeDefaults      = {
                fullMonths:
["January","February","March","April","May","June","July","August","September","Oc
tober","November","December"],
                monthAbbrs:
["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"],
                fullDays:
["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"],
                dayAbbrs:  ["Mon","Tue","Wed","Thu","Fri","Sat","Sun"],
                titles:    ["Previous month","Next month","Previous year","Next
year", "Today", "Show Calendar", "wk", "Week [[%0%]] of [[%1%]]", "Week", "Select
a date", "Click \u0026 Drag to move", "Display \u201C[[%0%]]\u201D first", "Go to
Today\u2019s date", "Disabled date"],
```

```
                firstDayOfWeek:0,
                imported:  false
            };

        void function() {
                var scriptFiles = document.getElementsByTagName('head')
[0].getElementsByTagName('script'),
                    scriptInner = scriptFiles[scriptFiles.length -
1].innerHTML.replace(/[\n\r\s\t]+/g, " ").replace(/^\s+/, "").replace(/\s+$/, ""),
                    json        = parseJSON(scriptInner);

                if(typeof json === "object" && !("err" in json)) {
                        affectJSON(json);
                };

                if(typeof(fdLocale) != "object") {
                        var loc    = scriptFiles[scriptFiles.length -
1].src.substr(0, scriptFiles[scriptFiles.length - 1].src.lastIndexOf("/")) +
"/lang/",
                            script;

                        for(var i = 0; i < languageInfo.length; i++) {
                                script = document.createElement('script');
                                script.type = "text/javascript";
                                script.src  = loc + languageInfo[i] + ".js";
                                script.setAttribute("charset", "utf-8");
                                /*@cc_on
                                /*@if(@_win32)
                                var bases = document.getElementsByTagName('base');
                                if (bases.length && bases[0].childNodes.length) {
                                        bases[0].appendChild(script);
                                } else {
                                        document.getElementsByTagName('head')
[0].appendChild(script);
                                };
                                bases = null;
                                @else @*/
                                document.getElementsByTagName('head')
[0].appendChild(script);
                                /*@end
                                @*/
                        };
                        script = null;
                } else {
                        returnLocaleDate = true;
                };

        }();

        function parseNavigatorLanguage() {
                var languageTag = navigator.language ?
navigator.language.toLowerCase() : navigator.userLanguage ?
navigator.userLanguage.toLowerCase() : "en";
```

```
                if(languageTag.search(/^([a-z]{2})-([a-z]{2})$/) != -1) {
                        return  [languageTag.match(/^([a-z]{2})-([a-z]{2})$/)[1],
languageTag];
                } else {
                        return [languageTag.substr(0,2)];
                };
        };

        function affectJSON(json) {
                if(typeof json !== "object") { return; };
                for(key in json) {
                        value = json[key];
                        switch(key.toLowerCase()) {
                                case "lang":
                                        if(value.search(/^[a-z]{2}(-([a-z]{2}))?
$/i) != -1) {
                                                languageInfo =
[value.toLowerCase()];
                                                returnLocaleDate = true;
                                        };
                                        break;
                                case "split":
                                        if(typeof value === 'object') {
                                                if(value.length && value.length ==
2) {
                                                        splitAppend = value;
                                                };
                                        };
                                        break;
                                case "formats":
                                        if(typeof value === 'object') {
                                                if(value.length) {
                                                        var tmpMasks = [];
                                                        for(var m = 0, msk; msk =
value[m]; m++) {
                                                                if(msk.match(/
((sp|dt|sl|ds|cc)|([d|D|l|j|N|w|S|W|M|F|m|n|t|Y|o|y|O|p]))(-((sp|dt|sl|ds|cc)|([d|
D|l|j|N|w|S|W|M|F|m|n|t|Y|o|y|O|p]))+/)) {
tmpMasks.push(msk);
                                                                };
                                                        };
                                                        if(tmpMasks.length) {
formatMasks = tmpMasks; };
                                                };
                                        };
                                        break;
                                case "nodrag":
                                        nodrag = !!value;
                                        break;
                                case "buttontabindex":
                                        buttonTabIndex = !!value;
                                        break;
                                case "cellformat":
```

```
                                        if(typeof value == "string" &&
value.match(/^((sp|dt|sl|ds|cc)|([d|D|l|j|N|w|S|W|M|F|m|n|t|Y|o|y|O|p]))(-((sp|dt|
sl|ds|cc)|([d|D|l|j|N|w|S|W|M|F|m|n|t|Y|o|y|O|p])))+$/)) {
                                                parseCellFormat(value);
                                        };
                                        break;
                                case "titleformat":
                                        if(value.match(/((sp|dt|sl|ds|cc)|([d|D|l|
j|N|w|S|W|M|F|m|n|t|Y|o|y|O|p]))(-((sp|dt|sl|ds|cc)|([d|D|l|j|N|w|S|W|M|F|m|n|t|Y|
o|y|O|p])))+/)) {
                                                titleFormat = value;
                                        };
                                };
                        };
                };

                function parseCellFormat(value) {
                        // I'm sure this could be done with a regExp and a split in one
line... seriously...
                        var parts       = value.split("-"),
                            fullParts   = [],
                            tmpParts    = [],
                            part;

                        for(var pt = 0; pt < parts.length; pt++) {
                                part = parts[pt];
                                if(part == "j" || part == "d") {
                                        if(tmpParts.length) {
                                                fullParts.push(tmpParts.join("-"));
                                                tmpParts = [];
                                        };
                                        fullParts.push("placeholder");
                                } else {
                                        tmpParts.push(part);
                                };
                        };

                        if(tmpParts.length) {
                                fullParts.push(tmpParts.join("-"));
                        };

                        if(!fullParts.length || fullParts.length > 3) {
                                formatParts = window.opera ? ["placeholder"] :
["placeholder", "sp-F-sp-Y"];
                                cellFormat = "j-sp-F-sp-Y";
                                return;
                        };

                        // Don't use hidden text for opera due to focus outline problems
                        formatParts = window.opera ? ["placeholder"] : fullParts;
                        cellFormat  = window.opera ? "j-sp-F-sp-Y" : value;
                };

                function pad(value, length) {
```

```
                length = length || 2;
                return "0000".substr(0,length - Math.min(String(value).length,
length)) + value;
        };

        function addEvent(obj, type, fn) {
                try {
                if( obj.attachEvent ) {
                        obj["e"+type+fn] = fn;
                        obj[type+fn] = function(){obj["e"+type+fn]( window.event
);};
                        obj.attachEvent( "on"+type, obj[type+fn] );
                } else {
                        obj.addEventListener( type, fn, true );
                };
                } catch(err) {
                        alert(obj + " " + type + " " + fn)
                }
        };

        function removeEvent(obj, type, fn) {
                try {
                        if( obj.detachEvent ) {
                                obj.detachEvent( "on"+type, obj[type+fn] );
                                obj[type+fn] = null;
                        } else {
                                obj.removeEventListener( type, fn, true );
                        };
                } catch(err) {};
        };

        function stopEvent(e) {
                e = e || document.parentWindow.event;
                if(e.stopPropagation) {
                        e.stopPropagation();
                        e.preventDefault();
                };
                /*@cc_on
                @if(@_win32)
                e.cancelBubble = true;
                e.returnValue = false;
                @end
                @*/
                return false;
        };

        function parseJSON(str) {
                // Check we have a String
                if(typeof str !== 'string' || str == "") { return {}; };
                try {
                        // Does a JSON (native or not) Object exist
                        if(typeof JSON === "object" && JSON.parse) {
                                return window.JSON.parse(str);
                        // Genious code taken from:
```

```
http://kentbrewster.com/badges/
                        } else if(/lang|split|formats|
nodrag/.test(str.toLowerCase())) {
                                var f = Function(['var
document,top,self,window,parent,Number,Date,Object,Function,',

'Array,String,Math,RegExp,Image,ActiveXObject;',
                                        'return (' , str.replace(/<\!--.+--
>/gim,'').replace(/\bfunction\b/g,'function') , ');'].join(''));
                                return f();
                        };
                } catch (e) { };
                return {"err":"Trouble parsing JSON object"};
        };

        function setARIARole(element, role) {
                if(element && element.tagName) {
                        element.setAttribute("role", role);
                };
        };

        function setARIAProperty(element, property, value) {
            if(element && element.tagName) {
                        element.setAttribute("aria-" + property, value);
                };
      };

        // The datePicker object itself
        function datePicker(options) {
                this.dateSet            = null;
                this.timerSet           = false;
                this.visible            = false;
                this.fadeTimer          = null;
                this.timer              = null;
                this.yearInc            = 0;
                this.monthInc           = 0;
                this.dayInc             = 0;
                this.mx                 = 0;
                this.my                 = 0;
                this.x                  = 0;
                this.y                  = 0;
                this.cursorDate         = options.cursorDate ? options.cursorDate
: "",
                this.date               = "cursorDate" in options &&
options.cursorDate ? new Date(+options.cursorDate.substr(0,4),
+options.cursorDate.substr(4,2) - 1, +options.cursorDate.substr(6,2)) : new
Date();
                this.defaults           = {};
                this.created            = false;
                this.disabled           = false;
                this.id                 = options.id;
                this.opacity            = 0;
                this.firstDayOfWeek     = localeImport.firstDayOfWeek;
                this.buttonWrapper      = "buttonWrapper" in options ?
```

```
options.buttonWrapper : false;
                this.staticPos          = "staticPos" in options ? !!
options.staticPos : false;
                this.disabledDays       = "disabledDays" in options &&
options.disabledDays.length ? options.disabledDays : [0,0,0,0,0,0,0];
                this.disabledDates      = "disabledDates" in options ?
options.disabledDates : {};
                this.enabledDates       = "enabledDates" in options ?
options.enabledDates : {};
                this.showWeeks          = "showWeeks" in options ? !!
options.showWeeks : false;
                this.low                = options.low || "";
                this.high               = options.high || "";
                this.dragDisabled       = nodrag ? true : ("dragDisabled" in
options ? !!options.dragDisabled : false);
                this.positioned         = "positioned" in options ?
options.positioned : false;
                this.hideInput          = this.staticPos ? false : "hideInput" in
options ? !!options.hideInput : false;
                this.splitDate          = "splitDate" in options ? !!
options.splitDate : false;
                this.format             = options.format || "d-sl-m-sl-Y";
                this.statusFormat       = options.statusFormat || "";
                this.highlightDays      = options.highlightDays &&
options.highlightDays.length ? options.highlightDays : [0,0,0,0,0,1,1];
                this.noFadeEffect       = "noFadeEffect" in options ? !!
options.noFadeEffect : false;
                this.opacityTo          = this.noFadeEffect || this.staticPos ?
99 : 90;
                this.callbacks          = {};
                this.fillGrid           = !!options.fillGrid;
                this.noToday            = !!options.noToday;
                this.labelledBy         =
findLabelForElement(document.getElementById(options.id));
                this.constrainSelection  = this.fillGrid && !!
options.constrainSelection;
                this.finalOpacity       = !this.staticPos && "finalOpacity" in
options ? +options.finalOpacity : 90;
                this.dynDisabledDates   = {};
                this.inUpdate           = false;
                this.noFocus            = true;
                this.kbEventsAdded      = false;
                this.fullCreate         = false;
                this.selectedTD         = null;
                this.cursorTD           = null;
                this.addSpans           = this.staticPos;
                this.spansAdded         = false;

                /*@cc_on
                @if(@_win32)
                this.interval           = new Date();
                this.iePopUp            = null;
                this.isIE7              = false;
                @end
```

```
                @*/

                /*@cc_on
                @if(@_jscript_version <= 5.7)
                this.isIE7              = document.documentElement && typeof
document.documentElement.style.maxHeight != "undefined";
                @end
                @*/

                for(var thing in options.callbacks) {
                        this.callbacks[thing] = options.callbacks[thing];
                };

                // Adjust time to stop daylight savings madness on windows
                this.date.setHours(12);

                this.changeHandler = function() {
                        o.setDateFromInput();
                        if(o.created) { o.updateTable(); };
                };
                this.getScrollOffsets = function() {
                        if(typeof(window.pageYOffset) == 'number') {
                                //Netscape compliant
                                return [window.pageXOffset, window.pageYOffset];
                        } else if(document.body && (document.body.scrollLeft ||
document.body.scrollTop)) {
                                //DOM compliant
                                return [document.body.scrollLeft,
document.body.scrollTop];
                        } else if(document.documentElement &&
(document.documentElement.scrollLeft || document.documentElement.scrollTop)) {
                                //IE6 standards compliant mode
                                return [document.documentElement.scrollLeft,
document.documentElement.scrollTop];
                        };
                        return [0,0];
                };
                this.reposition = function() {
                        if(!o.created || !o.getElem() || o.staticPos) { return; };

                        o.div.style.visibility = "hidden";
                        o.div.style.left = o.div.style.top = "0px";
                        o.div.style.display = "block";

                        var osh         = o.div.offsetHeight,
                            osw         = o.div.offsetWidth,
                            elem        = document.getElementById('fd-but-' +
o.id),

                            pos         = o.truePosition(elem),
                            trueBody    = (document.compatMode &&
document.compatMode!="BackCompat") ? document.documentElement : document.body,
                            sOffsets    = o.getScrollOffsets(),
                            scrollTop   = sOffsets[1],
                            scrollLeft  = sOffsets[0];
```

```
                        o.div.style.visibility = "visible";

                        o.div.style.left  =
Number(parseInt(trueBody.clientWidth+scrollLeft) < parseInt(osw+pos[0]) ?
Math.abs(parseInt((trueBody.clientWidth+scrollLeft) - osw)) : pos[0]) + "px";
                        o.div.style.top   =
Number(parseInt(trueBody.clientHeight+scrollTop) <
parseInt(osh+pos[1]+elem.offsetHeight+2) ? Math.abs(parseInt(pos[1] - (osh + 2)))
: Math.abs(parseInt(pos[1] + elem.offsetHeight + 2))) + "px";

                        /*@cc_on
                        @if(@_jscript_version <= 5.7)
                        if(o.isIE7) return;
                        o.iePopUp.style.top    = o.div.style.top;
                        o.iePopUp.style.left   = o.div.style.left;
                        o.iePopUp.style.width  = osw + "px";
                        o.iePopUp.style.height = (osh - 2) + "px";
                        @end
                        @*/
                };
                this.removeOldFocus = function() {
                        var td = document.getElementById(o.id + "-date-picker-
hover");
                        if(td) {
                                try {
                                        td.setAttribute(!/*@cc_on!@*/false ?
"tabIndex" : "tabindex", "-1");
                                        td.tabIndex = -1;
                                        td.className = td.className.replace(/date-
picker-hover/, "");
                                        td.id = "";
                                } catch(err) {};
                        };
                };
                this.setNewFocus = function() {
                        var td = document.getElementById(o.id + "-date-picker-
hover");
                        if(td) {
                                try {
                                        td.setAttribute(!/*@cc_on!@*/false ?
"tabIndex" : "tabindex", "0");
                                        td.tabIndex = 0;
                                        td.className = td.className.replace(/date-
picker-hover/, "") + " date-picker-hover";
                                        if(!this.noFocus) {
                                                setTimeout(function() { try {
td.focus(); } catch(err) {}; }, 0);
                                        };
                                } catch(err) {};
                        };
                };
                this.updateTable = function(noCallback) {
                        if(o.inUpdate) return;
```

```
                        o.inUpdate = true;
                        o.removeOldFocus();

                        if(o.timerSet) {
                                o.date.setDate(Math.min(o.date.getDate()+o.dayInc,
daysInMonth(o.date.getMonth()+o.monthInc,o.date.getFullYear()+o.yearInc)) );
                                o.date.setMonth(o.date.getMonth() + o.monthInc);
                                o.date.setFullYear(o.date.getFullYear() +
o.yearInc);
                        };

                        o.outOfRange();
                        if(!o.noToday) { o.disableTodayButton(); };
                        o.showHideButtons(o.date);

                        var cd = o.date.getDate(),
                            cm = o.date.getMonth(),
                            cy = o.date.getFullYear(),
                            cursorDate = (String(cy) + pad(cm+1) + pad(cd)),
                            tmpDate    = new Date(cy, cm, 1);

                        tmpDate.setHours(5);

                        var dt, cName, td, i, currentDate, cellAdded, col,
currentStub, abbr, bespokeRenderClass, spnC,
                        weekDayC            = ( tmpDate.getDay() + 6 ) % 7,
                        firstColIndex       = (((weekDayC - o.firstDayOfWeek) + 7
) % 7) - 1,
                        dpm                 = daysInMonth(cm, cy),
                        today               = new Date(),
                        dateSetD            = (o.dateSet != null) ?
o.dateSet.getFullYear() + pad(o.dateSet.getMonth()+1) + pad(o.dateSet.getDate()) :
false,
                        stub                = String(tmpDate.getFullYear()) +
pad(tmpDate.getMonth()+1),
                        cellAdded           = [4,4,4,4,4,4],
                        lm                  = new Date(cy, cm-1, 1),
                        nm                  = new Date(cy, cm+1, 1),
                        daySub              = daysInMonth(lm.getMonth(),
lm.getFullYear()),
                        stubN               = String(nm.getFullYear()) +
pad(nm.getMonth()+1),
                        stubP               = String(lm.getFullYear()) +
pad(lm.getMonth()+1),
                        weekDayN            = (nm.getDay() + 6) % 7,
                        weekDayP            = (lm.getDay() + 6) % 7,
                        today               = today.getFullYear() +
pad(today.getMonth()+1) + pad(today.getDate()),
                        spn                 = document.createElement('span'),
                        enabledDates        = o.getEnabledDates(cy, cm + 1);

                        o.firstDateShown    = !o.constrainSelection && o.fillGrid
&& (0 - firstColIndex < 1) ? String(stubP) + (daySub + (0 - firstColIndex)) : stub
```

```
+ "01";
                        o.lastDateShown     = !o.constrainSelection && o.fillGrid
? stubN + pad(41 - firstColIndex - dpm) : stub + String(dpm);
                        o.currentYYYYMM     = stub;

                        bespokeRenderClass  = o.callback("redraw", {id:o.id,
dd:pad(cd), mm:pad(cm+1), yyyy:cy, firstDateDisplayed:o.firstDateShown,
lastDateDisplayed:o.lastDateShown}) || {};
                        o.dynDisabledDates  = o.getDisabledDates(cy, cm + 1);

                        spn.className       = "fd-screen-reader";

                        if(this.selectedTD != null) {
                                setARIAProperty(this.selectedTD, "selected",
false);
                                this.selectedTD = null;
                        };

                        for(var curr = 0; curr < 42; curr++) {
                                row  = Math.floor(curr / 7);
                                td   = o.tds[curr];
                                spnC = spn.cloneNode(false);

                                while(td.firstChild)
td.removeChild(td.firstChild);

                                if((curr > firstColIndex && curr <= (firstColIndex
+ dpm)) || o.fillGrid) {
                                        currentStub     = stub;
                                        weekDay         = weekDayC;
                                        dt              = curr - firstColIndex;
                                        cName           = [];
                                        selectable      = true;

                                        if(dt < 1) {
                                                dt              = daySub + dt;
                                                currentStub     = stubP;
                                                weekDay         = weekDayP;
                                                selectable      = !
o.constrainSelection;
                                                cName.push("month-out");
                                        } else if(dt > dpm) {
                                                dt -= dpm;
                                                currentStub     = stubN;
                                                weekDay         = weekDayN;
                                                selectable      = !
o.constrainSelection;
                                                cName.push("month-out");
                                        };

                                        weekDay = ( weekDay + dt + 6 ) % 7;
                                        cName.push("day-" +
localeDefaults.dayAbbrs[weekDay].toLowerCase());
```

```
                                        currentDate = currentStub + String(dt < 10
? "0" : "") + dt;

                                        if(o.low && +currentDate < +o.low ||
o.high && +currentDate > +o.high) {
                                                td.className = "out-of-range";
                                                td.title = "";

td.appendChild(document.createTextNode(dt));
                                                if(o.showWeeks) { cellAdded[row] =
Math.min(cellAdded[row], 2); };
                                        } else {
                                                if(selectable) {
                                                        td.title = titleFormat ?
printFormattedDate(new Date(+String(currentStub).substr(0,4),
+String(currentStub).substr(4, 2) - 1, +dt), titleFormat, true) : "";
                                                        cName.push("cd-" +
currentDate + " yyyymm-" + currentStub + " mmdd-" + currentStub.substr(4,2) +
pad(dt));
                                                } else {
                                                        td.title = titleFormat ?
getTitleTranslation(13) + " " + printFormattedDate(new
Date(+String(currentStub).substr(0,4), +String(currentStub).substr(4, 2) - 1,
+dt), titleFormat, true) : "";
                                                        cName.push("yyyymm-" +
currentStub + " mmdd-" + currentStub.substr(4,2) + pad(dt) + " not-selectable");
                                                };

                                                if(currentDate == today) {
cName.push("date-picker-today"); };

                                                if(dateSetD == currentDate) {
                                                        cName.push("date-picker-
selected-date");
                                                        setARIAProperty(td,
"selected", "true");
                                                        this.selectedTD = td;
                                                };

                                                if((o.disabledDays[weekDay] && !
(currentDate in enabledDates)) || currentDate in o.dynDisabledDates) {
cName.push("day-disabled"); if(titleFormat && selectable) { td.title =
getTitleTranslation(13) + " " + td.title; }; }

                                                if(currentDate in
bespokeRenderClass) { cName.push(bespokeRenderClass[currentDate]); }

                                                if(o.highlightDays[weekDay]) {
cName.push("date-picker-highlight"); };

                                                if(cursorDate == currentDate) {
                                                        td.id = o.id + "-date-
picker-hover";
                                                };
```

```
                                                td.className = cName.join(" ");

                                                if(o.kbEventsAdded || o.addSpans)
{
                                                        for(var pt = 0, part; part
= formatParts[pt]; pt++) {
                                                                if(part ==
"placeholder") {

td.appendChild(document.createTextNode(dt));
                                                                } else {
                                                                        spnC =
spn.cloneNode(spn);

spnC.appendChild(document.createTextNode(printFormattedDate(new
Date(+String(currentStub).substr(0,4), +String(currentStub).substr(4, 2) - 1,
+dt), part, true)));

td.appendChild(spnC);
                                                                };
                                                        };

                                                } else {

td.appendChild(document.createTextNode(dt));
                                                };

                                                if(o.showWeeks) {
                                                        cellAdded[row] =
Math.min(cName[0] == "month-out" ? 3 : 1, cellAdded[row]);
                                                };
                                        };
                                } else {
                                        td.className = "date-picker-unused";

td.appendChild(document.createTextNode(nbsp));
                                        td.title = "";
                                };

                                if(o.showWeeks && curr - (row * 7) == 6) {
                                        while(o.wkThs[row].firstChild)
o.wkThs[row].removeChild(o.wkThs[row].firstChild);

o.wkThs[row].appendChild(document.createTextNode(cellAdded[row] == 4 && !
o.fillGrid ? nbsp : getWeekNumber(cy, cm, curr - firstColIndex - 6)));
                                        o.wkThs[row].className = "date-picker-
week-header" + (["","","" out-of-range","" month-out",""][cellAdded[row]]);
                                };
                        };

                        o.spansAdded = o.kbEventsAdded || o.addSpans;
                        var span = o.titleBar.getElementsByTagName("span");
                        while(span[0].firstChild)
```

```
span[0].removeChild(span[0].firstChild);
                        while(span[1].firstChild)
span[1].removeChild(span[1].firstChild);

span[0].appendChild(document.createTextNode(getMonthTranslation(cm, false) +
nbsp));
                        span[1].appendChild(document.createTextNode(cy));

                        if(o.timerSet) {
                                o.timerInc = 50 + Math.round(((o.timerInc - 50) /
1.8));
                                o.timer = window.setTimeout(o.updateTable,
o.timerInc);
                        };

                        o.inUpdate = false;
                        o.setNewFocus();
                };

                this.show = function(autoFocus) {
                        if(this.staticPos) { return; };

                        var elem = this.getElem();
                        if(!elem || (elem && elem.disabled)) { return; };

                        this.noFocus = true;
                        if(!document.getElementById('fd-' + this.id)) {
                                this.created    = false;
                                this.fullCreate = false;
                                this.create();
                                this.fullCreate = true;
                        } else {
                                this.setDateFromInput();
                                this.reposition();
                        };

                        this.noFocus = !!!autoFocus;

                        if(this.noFocus) {
                                addEvent(document, "mousedown", this.onmousedown);
                        };

                        this.updateTable();

                        this.opacityTo = this.finalOpacity;
                        this.div.style.display = "block";

                        /*@cc_on
                        @if(@_jscript_version <= 5.7)

                        if(!o.isIE7) {
                                this.iePopUp.style.width = this.div.offsetWidth +
"px";
                                this.iePopUp.style.height = this.div.offsetHeight
```

```
+ "px";
                                        this.iePopUp.style.display = "block";
                                };

                                @end
                                @*/

                                this.setNewFocus();
                                this.fade();
                                var butt = document.getElementById('fd-but-' + this.id);
                                if(butt) { butt.className = butt.className.replace("dp-
button-active", "") + " dp-button-active"; };
                        };
                        this.hide = function() {
                                if(!this.visible || !this.created || !
document.getElementById('fd-' + this.id)) return;

                                this.stopTimer();
                                this.removeFocusEvents();

                                if(this.staticPos) { return; };

                                var butt = document.getElementById('fd-but-' + this.id);
                                if(butt) butt.className = butt.className.replace("dp-
button-active", "");

                                removeEvent(document, "mousedown", this.onmousedown);

                                /*@cc_on
                                @if(@_jscript_version <= 5.7)
                                if(!this.isIE7) { this.iePopUp.style.display = "none"; };
                                @end
                                @*/

                                this.opacityTo = 0;
                                this.fade();

                                // Programmatically remove the focus style on the calendar
                                this.div.className =
this.div.className.replace("datepicker-focus", "");

                                // Update status bar
                                if(this.statusBar) {
this.updateStatus(getTitleTranslation(9)); };
                        };
                        this.destroy = function() {

                                if(document.getElementById("fd-but-" + this.id)) {
                                        document.getElementById("fd-but-" +
this.id).parentNode.removeChild(document.getElementById("fd-but-" + this.id));
                                };

                                if(!this.created) { return; };
```

```
                        // Cleanup for Internet Explorer
                        removeEvent(this.table, "mousedown", o.onmousedown);
                        removeEvent(this.table, "mouseover", o.onmouseover);
                        removeEvent(this.table, "mouseout", o.onmouseout);
                        removeEvent(document, "mousedown", o.onmousedown);
                        removeEvent(document, "mouseup",   o.clearTimer);
                        o.removeFocusEvents();
                        clearTimeout(o.fadeTimer);
                        clearTimeout(o.timer);

                        /*@cc_on
                        @if(@_jscript_version <= 5.7)
                        if(!o.staticPos && !o.isIE7) {
                                try {

o.iePopUp.parentNode.removeChild(o.iePopUp);
                                        o.iePopUp = null;
                                } catch(err) {};
                        };
                        @end
                        @*/

                        if(this.div && this.div.parentNode) {
                                this.div.parentNode.removeChild(this.div);
                        };

                        o = null;
                };
                this.resizeInlineDiv = function()  {
                        o.div.style.width = o.table.offsetWidth + "px";
                        o.div.style.height = o.table.offsetHeight + "px";
                };
                this.create = function() {
                        if(this.created) { return; };

                        this.noFocus = true;

                        function createTH(details) {
                                var th = document.createElement('th');
                                if(details.thClassName) th.className =
details.thClassName;
                                if(details.colspan) {
                                        /*@cc_on
                                        /*@if (@_win32)

th.setAttribute('colSpan',details.colspan);
                                        @else @*/

th.setAttribute('colspan',details.colspan);
                                        /*@end
                                        @*/
                                };
                                /*@cc_on
                                /*@if (@_win32)
```

```
                        th.unselectable = "on";
                        /*@end@*/
                        return th;
                };
                function createThAndButton(tr, obj) {
                        for(var i = 0, details; details = obj[i]; i++) {
                                var th = createTH(details);
                                tr.appendChild(th);
                                var but = document.createElement('span');
                                but.className = details.className;
                                but.id = o.id + details.id;

but.appendChild(document.createTextNode(details.text || o.nbsp));
                                but.title = details.title || "";
                                /*@cc_on
                                /*@if(@_win32)
                                th.unselectable = but.unselectable = "on";
                                /*@end@*/
                                th.appendChild(but);
                        };
                };

                this.div                    =
document.createElement('div');
                this.div.id                 = "fd-" + this.id;
                this.div.className          = "datePicker";

                // Attempt to hide the div from screen readers during
content creation
                this.div.style.visibility = "hidden";
                this.div.style.display = "none";

                // Set the ARIA describedby property if the required block
available
                if(document.getElementById("fd-datepicker-aria-
describedby")) {
                        setARIAProperty(this.div, "describedby", "fd-
datepicker-aria-describedby");
                };

                // Set the ARIA labelled property if the required label
available
                if(this.labelledBy) {
                        setARIAProperty(this.div, "labelledby",
this.labelledBy.id);
                };

                var tr, row, col, tableHead, tableBody, tableFoot;

                this.table            = document.createElement('table');
                this.table.className  = "datePickerTable";
                this.table.onmouseover = this.onmouseover;
                this.table.onmouseout  = this.onmouseout;
                this.table.onclick    = this.onclick;
```

```
                        if(this.staticPos) {
                                this.table.onmousedown  = this.onmousedown;
                        };

                        this.div.appendChild(this.table);

                        var dragEnabledCN = !this.dragDisabled ? " drag-enabled" :
"";

                        if(!this.staticPos) {
                                this.div.style.visibility = "hidden";
                                this.div.className += dragEnabledCN;
                                document.getElementsByTagName('body')
[0].appendChild(this.div);

                                /*@cc_on
                                @if(@_jscript_version <= 5.7)

                                if(!this.isIE7) {
                                        this.iePopUp =
document.createElement('iframe');
                                        this.iePopUp.src =
"javascript:'<html></html>';";

this.iePopUp.setAttribute('className','iehack');
                                        // Remove iFrame from tabIndex
                                  this.iePopUp.setAttribute("tabIndex", -1);

                                        // Hide it from ARIA aware technologies
                                  setARIARole(this.iePopUp, "presentation");
                                        setARIAProperty(this.iePopUp, "hidden",
"true");
                                        this.iePopUp.scrolling = "no";
                                        this.iePopUp.frameBorder = "0";
                                        this.iePopUp.name = this.iePopUp.id =
this.id + "-iePopUpHack";
                                        document.body.appendChild(this.iePopUp);
                                };

                                @end
                                @*/

                                // Aria "hidden" property for non active popup
datepickers
                                setARIAProperty(this.div, "hidden", "true");
                        } else {
                                elem = this.positioned ?
document.getElementById(this.positioned) : this.getElem();
                                if(!elem) {
                                        this.div = null;
                                        throw this.positioned ? "Could not locate
a datePickers associated parent element with an id:" + this.positioned : "Could
not locate a datePickers associated input with an id:" + this.id;
```

```
                                };

                                this.div.className += " static-datepicker";

                                if(this.positioned) {
                                        elem.appendChild(this.div);
                                } else {
                                        elem.parentNode.insertBefore(this.div,
elem.nextSibling);
                                };

                                if(this.hideInput) {
                                        var elemList = [elem];
                                        if(this.splitDate) {
                                                elemList[elemList.length] =
document.getElementById(this.id + splitAppend[1]);
                                                elemList[elemList.length] =
document.getElementById(this.id + splitAppend[0]);
                                        };
                                        for(var i = 0; i < elemList.length; i++) {
                                                if(elemList[i].tagName)
elemList[i].className += " fd-hidden-input";
                                        };
                                };

                                setTimeout(this.resizeInlineDiv, 300);
                        };

                        // ARIA Grid role
                        setARIARole(this.div, "grid");

                        if(this.statusFormat) {
                                tableFoot = document.createElement('tfoot');
                                this.table.appendChild(tableFoot);
                                tr = document.createElement('tr');
                                tr.className = "date-picker-tfoot";
                                tableFoot.appendChild(tr);
                                this.statusBar = createTH({thClassName:"date-
picker-statusbar" + dragEnabledCN, colspan:this.showWeeks ? 8 : 7});
                                tr.appendChild(this.statusBar);
                                this.updateStatus();
                        };

                        tableHead = document.createElement('thead');
                        this.table.appendChild(tableHead);

                        tr  = document.createElement('tr');
                        setARIARole(tr, "presentation");

                        tableHead.appendChild(tr);

                        // Title Bar
                        this.titleBar = createTH({thClassName:"date-picker-title"
+ dragEnabledCN, colspan:this.showWeeks ? 8 : 7});
```

```
                        tr.appendChild(this.titleBar);
                        tr = null;

                        var span = document.createElement('span');
                        span.appendChild(document.createTextNode(nbsp));
                        span.className = "month-display" + dragEnabledCN;
                        this.titleBar.appendChild(span);

                        span = document.createElement('span');
                        span.appendChild(document.createTextNode(nbsp));
                        span.className = "year-display" + dragEnabledCN;
                        this.titleBar.appendChild(span);

                        span = null;

                        tr  = document.createElement('tr');
                        setARIARole(tr, "presentation");
                        tableHead.appendChild(tr);

                        createThAndButton(tr, [
                        {className:"prev-but prev-year",  id:"-prev-year-but",
text:"\u00AB", title:getTitleTranslation(2) },
                        {className:"prev-but prev-month", id:"-prev-month-but",
text:"\u2039", title:getTitleTranslation(0) },
                        {colspan:this.showWeeks ? 4 : 3, className:"today-but",
id:"-today-but", text:getTitleTranslation(4)},
                        {className:"next-but next-month", id:"-next-month-but",
text:"\u203A", title:getTitleTranslation(1)},
                        {className:"next-but next-year",  id:"-next-year-but",
text:"\u00BB", title:getTitleTranslation(3) }
                        ]);

                        tableBody = document.createElement('tbody');
                        this.table.appendChild(tableBody);

                        var colspanTotal = this.showWeeks ? 8 : 7,
                            colOffset    = this.showWeeks ? 0 : -1,
                            but, abbr;

                        for(var rows = 0; rows < 7; rows++) {
                                row = document.createElement('tr');

                                if(rows != 0) {
                                        // ARIA Grid role
                                        setARIARole(row, "row");
                                        tableBody.appendChild(row);
                                } else {
                                        tableHead.appendChild(row);
                                };

                                for(var cols = 0; cols < colspanTotal; cols++) {
                                        if(rows === 0 || (this.showWeeks && cols
=== 0)) {
```

```
                                                col =
document.createElement('th');
                                        } else {
                                                col =
document.createElement('td');

                                                col.onblur  = this.onblur;
                                                col.onfocus = this.onfocus;
                                                setARIAProperty(col,
"describedby", this.id + "-col-" + cols + (this.showWeeks ? " " + this.id + "-
row-" + rows : ""));
                                                setARIAProperty(col, "selected",
"false");
                                        };

                                        /*@cc_on@*/
                                        /*@if(@_win32)
                                        col.unselectable = "on";
                                        /*@end@*/

                                        row.appendChild(col);
                                        if((this.showWeeks && cols > 0 && rows >
0) || (!this.showWeeks && rows > 0)) {
                                                setARIARole(col, "gridcell");
                                        } else {
                                                if(rows === 0 && cols > colOffset)
{
                                                        col.className = "date-
picker-day-header";

                                                        col.scope = "col";
                                                        setARIARole(col,
"columnheader");

                                                        col.id = this.id + "-col-"
+ cols;
                                                } else {
                                                        col.className = "date-
picker-week-header";

                                                        col.scope = "row";
                                                        setARIARole(col,
"rowheader");

                                                        col.id = this.id + "-row-"
+ rows;
                                                };
                                        };
                                };
                        };

                        col = row = null;

                        this.ths = this.table.getElementsByTagName('thead')
[0].getElementsByTagName('tr')[2].getElementsByTagName('th');
                        for (var y = 0; y < colspanTotal; y++) {
                                if(y == 0 && this.showWeeks) {
```

```
this.ths[y].appendChild(document.createTextNode(getTitleTranslation(6)));
                                        this.ths[y].title =
getTitleTranslation(8);
                                        continue;
                                };

                                if(y > (this.showWeeks ? 0 : -1)) {
                                        but = document.createElement("span");
                                        but.className = "fd-day-header";
                                        /*@cc_on@*/
                                        /*@if(@_win32)
                                        but.unselectable = "on";
                                        /*@end@*/
                                        this.ths[y].appendChild(but);
                                };
                        };

                        but = null;

                        this.trs             =
this.table.getElementsByTagName('tbody')[0].getElementsByTagName('tr');
                        this.tds             =
this.table.getElementsByTagName('tbody')[0].getElementsByTagName('td');
                        this.butPrevYear     = document.getElementById(this.id +
"-prev-year-but");
                        this.butPrevMonth    = document.getElementById(this.id +
"-prev-month-but");
                        this.butToday        = document.getElementById(this.id +
"-today-but");
                        this.butNextYear     = document.getElementById(this.id +
"-next-year-but");
                        this.butNextMonth    = document.getElementById(this.id +
"-next-month-but");

                        if(this.noToday) {
                                this.butToday.style.display = "none";
                        };

                        if(this.showWeeks) {
                                this.wkThs =
this.table.getElementsByTagName('tbody')[0].getElementsByTagName('th');
                                this.div.className += " weeks-displayed";
                        };

                        tableBody = tableHead = tr = createThAndButton = createTH
= null;

                        if(this.low && this.high && (this.high - this.low < 7)) {
this.equaliseDates(); };

                        this.setDateFromInput();
                        this.updateTableHeaders();
                        this.created = true;
                        this.callback("create", {id:this.id});
```

```
                        this.updateTable();

                        if(this.staticPos) {
                                this.visible = true;
                                this.opacity = this.opacityTo;
                                this.div.style.visibility = "visible";
                                this.div.style.display = "block";
                                this.noFocus = true;
                                this.fade();
                        } else {
                                this.reposition();
                                this.div.style.visibility = "visible";
                                this.fade();
                                this.noFocus = true;
                        };

                        this.addSpans = false;
                };
                this.fade = function() {
                        window.clearTimeout(o.fadeTimer);
                        o.fadeTimer = null;
                        var diff = Math.round(o.opacity + ((o.opacityTo -
o.opacity) / 4));
                        o.setOpacity(diff);
                        if(Math.abs(o.opacityTo - diff) > 3 && !o.noFadeEffect) {
                                o.fadeTimer = window.setTimeout(o.fade, 50);
                        } else {
                                o.setOpacity(o.opacityTo);
                                if(o.opacityTo == 0) {
                                        o.div.style.display    = "none";
                                        o.div.style.visibility = "hidden";
                                        setARIAProperty(o.div, "hidden", "true");
                                        o.visible = false;
                                } else {
                                        setARIAProperty(o.div, "hidden", "false");
                                        o.visible = true;
                                };
                        };
                };
                this.trackDrag = function(e) {
                        e = e || window.event;
                        var diffx = (e.pageX?e.pageX:e.clientX?e.clientX:e.x) -
o.mx;
                        var diffy = (e.pageY?e.pageY:e.clientY?e.clientY:e.Y) -
o.my;
                        o.div.style.left = Math.round(o.x + diffx) > 0 ?
Math.round(o.x + diffx) + 'px' : "0px";
                        o.div.style.top  = Math.round(o.y + diffy) > 0 ?
Math.round(o.y + diffy) + 'px' : "0px";
                        /*@cc_on
                        @if(@_jscript_version <= 5.7)
                        if(o.staticPos || o.isIE7) return;
                        o.iePopUp.style.top    = o.div.style.top;
                        o.iePopUp.style.left   = o.div.style.left;
```

```
                        @end
                        @*/
                };
                this.stopDrag = function(e) {
                        removeEvent(document,'mousemove',o.trackDrag, false);
                        removeEvent(document,'mouseup',o.stopDrag, false);
                        o.div.style.zIndex = 9999;
                };
                this.onmousedown = function(e) {
                        e = e || document.parentWindow.event;
                        var el     = e.target != null ? e.target : e.srcElement,
                            origEl = el,
                            hideDP = true,
                            reg    = new RegExp("^fd-(but-)?" + o.id + "$");

                        o.mouseDownElem = null;

                        // Are we within the wrapper div or the button
                        while(el) {
                                if(el.id && el.id.length && el.id.search(reg) !=
-1) {
                                        hideDP = false;
                                        break;
                                };
                                try { el = el.parentNode; } catch(err) { break; };
                        };

                        // If not, then ...
                        if(hideDP) {
                                hideAll();
                                return true;
                        };

                        if((o.div.className + origEl.className).search('fd-
disabled') != -1) { return true; };

                        // We check the mousedown events on the buttons
                        if(origEl.id.search(new RegExp("^" + o.id + "(-prev-year-
but|-prev-month-but|-next-month-but|-next-year-but)$")) != -1) {
                                o.mouseDownElem = origEl;

                                addEvent(document, "mouseup", o.clearTimer);
                                addEvent(origEl, "mouseout",  o.clearTimer);

                                var incs = {
                                        "-prev-year-but":[0,-1,0],
                                        "-prev-month-but":[0,0,-1],
                                        "-next-year-but":[0,1,0],
                                        "-next-month-but":[0,0,1]
                                    },
                                    check = origEl.id.replace(o.id, ""),
                                    dateYYYYMM = Number(o.date.getFullYear() +
pad(o.date.getMonth()+1));
```

```
                                o.timerInc      = (o.currentYYYYMM > dateYYYYMM ||
o.currentYYYYMM < dateYYYYMM) ? 1600 : 800;
                                o.timerSet      = true;
                                o.dayInc        = incs[check][0];
                                o.yearInc       = incs[check][1];
                                o.monthInc      = incs[check][2];

                                o.addSpans      = false;

                                o.updateTable();

                        } else if(el.className.search("drag-enabled") != -1) {
                                o.mx = e.pageX ? e.pageX : e.clientX ? e.clientX :
e.x;
                                o.my = e.pageY ? e.pageY : e.clientY ? e.clientY :
e.Y;
                                o.x  = parseInt(o.div.style.left);
                                o.y  = parseInt(o.div.style.top);
                                addEvent(document,'mousemove',o.trackDrag, false);
                                addEvent(document,'mouseup',o.stopDrag, false);
                                o.div.style.zIndex = 10000;
                        };
                        return true;
                };
                this.onclick = function(e) {
                        if(o.opacity != o.opacityTo || o.disabled) return
stopEvent(e);

                        e = e || document.parentWindow.event;
                        var el = e.target != null ? e.target : e.srcElement;

                        while(el.parentNode) {
                                // Are we within a valid TD node
                                if(el.tagName && el.tagName.toLowerCase() == "td")
{
                                        if(el.className.search(/cd-([0-9]{8})/) ==
-1 || el.className.search(/date-picker-unused|out-of-range|day-disabled|no-
selection|not-selectable/) != -1) return stopEvent(e);
                                        var cellDate = el.className.match(/cd-([0-
9]{8})/)[1];
                                        o.date          = new
Date(cellDate.substr(0,4),cellDate.substr(4,2)-1,cellDate.substr(6,2));
                                        o.dateSet       = new Date(o.date);
                                        o.noFocus       = true;
                                        o.returnFormattedDate();
                                        o.hide();
                                        o.stopTimer();
                                        break;
                                } else if(el.id && el.id == o.id + "-today-but") {
                                        o.date = new Date();
                                        o.updateTable();
                                        o.stopTimer();
                                        break;
                                } else if(el.className.search(/date-picker-day-
```

```
header/) != -1) {
                                        var cnt = o.showWeeks ? -1 : 0,
                                        elem = el;

                                        while(elem.previousSibling) {
                                                elem = elem.previousSibling;
                                                if(elem.tagName.toLowerCase() ==
"th") cnt++;
                                        };

                                        o.firstDayOfWeek = (o.firstDayOfWeek +
cnt) % 7;
                                        o.updateTableHeaders();
                                        break;
                                };
                                try { el = el.parentNode; } catch(err) { break; };
                        };

                        return stopEvent(e);
                };
                this.onblur = function(e) {
                        e = e || document.parentWindow.event;
                        var el = e.target != null ? e.target : e.srcElement;

                        while(el.parentNode) {
                                if(el.id && el.id == "fd-" + o.id) {
                                        return true;
                                };
                                try { el = el.parentNode; } catch(err) { break; };
                        };

                        // Remove the keyboard events from the document
                        if(o.kbEventsAdded) o.removeFocusEvents();

                        // Programmatically remove the focus style on the calendar
                        o.div.className = o.div.className.replace("datepicker-
focus", "");
                        o.noFocus  = true;
                        o.addSpans = false;
                        o.hide();

                        // Update status bar
                        if(o.statusBar) { o.updateStatus(getTitleTranslation(9));
};
                };
                this.onfocus = function(e) {
                        if(o.staticPos) {
                                o.noFocus = false;
                                if(!o.spansAdded) {
                                        o.addSpans = true;
                                        o.updateTable();
                                };
                        };
                        o.addFocusEvents();
```

```
                        //o.noFocus = false;
                };
                this.onkeydown = function (e) {
                        o.stopTimer();
                        if(!o.visible) return false;

                        if(e == null) e = document.parentWindow.event;
                        var kc = e.keyCode ? e.keyCode : e.charCode;

                        if( kc == 13 ) {
                                // RETURN/ENTER: close & select the date
                                var td = document.getElementById(o.id + "-date-
picker-hover");
                                if(!td || td.className.search(/cd-([0-9]{8})/) ==
-1 || td.className.search(/no-selection|out-of-range|day-disabled/) != -1) {
                                        return stopEvent(e);
                                };
                                o.dateSet = new Date(o.date);
                                o.returnFormattedDate();
                                o.hide();
                                return stopEvent(e);
                        } else if(kc == 27) {
                                // ESC: close, no date selection
                                o.hide();
                                return stopEvent(e);
                        } else if(kc == 32 || kc == 0) {
                                // SPACE: goto today's date
                                o.date = new Date();
                                o.updateTable();
                                return stopEvent(e);
                        };

                        // Internet Explorer fires the keydown event faster than
the JavaScript engine can
                        // update the interface. The following attempts to fix
this.

                        /*@cc_on
                        @if(@_win32)
                        if(new Date().getTime() - o.interval.getTime() < 50)
{ return stopEvent(e); };
                        o.interval = new Date();
                        @end
                        @*/

                        if ((kc > 49 && kc < 56) || (kc > 97 && kc < 104)) {
                                if(kc > 96) kc -= (96-48);
                                kc -= 49;
                                o.firstDayOfWeek = (o.firstDayOfWeek + kc) % 7;
                                o.updateTableHeaders();
                                return stopEvent(e);
                        };

                        if ( kc < 33 || kc > 40 ) return true;
```

```
                        var d = new Date(o.date), tmp, cursorYYYYMM =
o.date.getFullYear() + pad(o.date.getMonth()+1);

                        // HOME: Set date to first day of current month
                        if(kc == 36) {
                                d.setDate(1);
                        // END: Set date to last day of current month
                        } else if(kc == 35) {

d.setDate(daysInMonth(d.getMonth(),d.getFullYear()));
                        // PAGE UP & DOWN
                        } else if ( kc == 33 || kc == 34) {
                                var add = (kc == 34) ? 1 : -1;

                                // CTRL + PAGE UP/DOWN: Moves to the same date in
the previous/next year
                                if(e.ctrlKey) {
                                        d.setFullYear(d.getFullYear() + add);
                                // PAGE UP/DOWN: Moves to the same date in the
previous/next month
                                } else {
                                        if(!((kc == 33 && o.currentYYYYMM >
cursorYYYYMM) || (kc == 34 && o.currentYYYYMM < cursorYYYYMM))) {
                                                tmp = new Date(d);
                                                tmp.setDate(2);
                                                tmp.setMonth(d.getMonth() + add);
                                                d.setDate(Math.min(d.getDate(),
daysInMonth(tmp.getMonth(),tmp.getFullYear())));
                                                d.setMonth(d.getMonth() + add);
                                        };
                                };
                        // LEFT ARROW
                        } else if ( kc == 37 ) {
                                d = new Date(o.date.getFullYear(),
o.date.getMonth(), o.date.getDate() - 1);
                        // RIGHT ARROW
                        } else if ( kc == 39 || kc == 34) {
                                d = new Date(o.date.getFullYear(),
o.date.getMonth(), o.date.getDate() + 1 );
                        // UP ARROW
                        } else if ( kc == 38 ) {
                                d = new Date(o.date.getFullYear(),
o.date.getMonth(), o.date.getDate() - 7);
                        // DOWN ARROW
                        } else if ( kc == 40 ) {
                                d = new Date(o.date.getFullYear(),
o.date.getMonth(), o.date.getDate() + 7);
                        };

                        if(o.outOfRange(d)) { return stopEvent(e); };
                        o.date = d;

                        if(o.statusBar) {
```

```
o.updateStatus(printFormattedDate(o.date, o.statusFormat, true)); };
                        var t = String(o.date.getFullYear()) +
pad(o.date.getMonth()+1) + pad(o.date.getDate());

                        if(e.ctrlKey || (kc == 33 || kc == 34) || t <
o.firstDateShown || t > o.lastDateShown) {
                                o.updateTable();
                                /*@cc_on
                                @if(@_win32)
                                o.interval = new Date();
                                @end
                                @*/
                        } else {
                                if(!o.noToday) { o.disableTodayButton(); };
                                o.removeOldFocus();
                                var dt = "cd-" + o.date.getFullYear() +
pad(o.date.getMonth()+1) + pad(o.date.getDate());

                                for(var i = 0, td; td = o.tds[i]; i++) {
                                        if(td.className.search(dt) == -1) {
                                                continue;
                                        };
                                        o.showHideButtons(o.date);
                                        td.id = o.id + "-date-picker-hover";
                                        o.setNewFocus();
                                        break;
                                };
                        };

                        return stopEvent(e);
                };
                this.onmouseout = function(e) {
                        e = e || document.parentWindow.event;
                        var p = e.toElement || e.relatedTarget;
                        while (p && p != this) try { p = p.parentNode} catch(e) {
p = this; };
                        if (p == this) return false;
                        if(o.currentTR) {
                                o.currentTR.className = "";
                                o.currentTR = null;
                        };
                        if(o.statusBar) {
o.updateStatus(printFormattedDate(o.date, o.statusFormat, true)); };
                };
                this.onmouseover = function(e) {
                        e = e || document.parentWindow.event;
                        var el = e.target != null ? e.target : e.srcElement;
                        while(el.nodeType != 1) { el = el.parentNode; };

                        if(!el || ! el.tagName) { return; };

                        var statusText = getTitleTranslation(9);
                        switch (el.tagName.toLowerCase()) {
                                case "td":
```

```
                                        if(el.className.search(/date-picker-
unused|out-of-range/) != -1) {
                                                statusText =
getTitleTranslation(9);
                                        } if(el.className.search(/cd-([0-9]{8})/)
!= -1) {
                                                o.stopTimer();
                                                var cellDate =
el.className.match(/cd-([0-9]{8})/)[1];

                                                o.removeOldFocus();
                                                el.id = o.id+"-date-picker-hover";
                                                o.setNewFocus();

                                                o.date = new
Date(+cellDate.substr(0,4),+cellDate.substr(4,2)-1,+cellDate.substr(6,2));
                                                if(!o.noToday) {
o.disableTodayButton(); };
                                                statusText =
printFormattedDate(o.date, o.statusFormat, true);
                                        };
                                        break;
                                case "th":
                                        if(!o.statusBar) { break; };
                                        if(el.className.search(/drag-enabled/) !=
-1) {
                                                statusText =
getTitleTranslation(10);
                                        } else if(el.className.search(/date-
picker-week-header/) != -1) {
                                                var txt = el.firstChild ?
el.firstChild.nodeValue : "";
                                                statusText = txt.search(/^(\d+)$/)
!= -1 ? getTitleTranslation(7, [txt, txt < 3 && o.date.getMonth() == 11 ?
getWeeksInYear(o.date.getFullYear()) + 1 : getWeeksInYear(o.date.getFullYear())])
: getTitleTranslation(9);
                                        };
                                        break;
                                case "span":
                                        if(!o.statusBar) { break; };
                                        if(el.className.search(/drag-enabled/) !=
-1) {
                                                statusText =
getTitleTranslation(10);
                                        } else if(el.className.search(/day-([0-
6])/) != -1) {
                                                var day = el.className.match(/day-
([0-6])/)[1];
                                                statusText =
getTitleTranslation(11, [getDayTranslation(day, false)]);
                                        } else if(el.className.search(/prev-year/)
!= -1) {
                                                statusText =
getTitleTranslation(2);
```

```
                                        } else if(el.className.search(/prev-
month/) != -1) {
                                                statusText =
getTitleTranslation(0);
                                        } else if(el.className.search(/next-year/)
!= -1) {
                                                statusText =
getTitleTranslation(3);
                                        } else if(el.className.search(/next-
month/) != -1) {
                                                statusText =
getTitleTranslation(1);
                                        } else if(el.className.search(/today-but/)
!= -1 && el.className.search(/disabled/) == -1) {
                                                statusText =
getTitleTranslation(12);
                                        };
                                        break;
                                default:
                                        statusText = "";
                        };
                        while(el.parentNode) {
                                el = el.parentNode;
                                if(el.nodeType == 1 && el.tagName.toLowerCase() ==
"tr") {
                                        if(o.currentTR) {
                                                if(el == o.currentTR) break;
                                                o.currentTR.className = "";
                                        };
                                        el.className = "dp-row-highlight";
                                        o.currentTR = el;
                                        break;
                                };
                        };
                        if(o.statusBar && statusText) {
o.updateStatus(statusText); };
                };
                this.clearTimer = function() {
                        o.stopTimer();
                        o.timerInc      = 800;
                        o.yearInc       = 0;
                        o.monthInc      = 0;
                        o.dayInc        = 0;

                        removeEvent(document, "mouseup", o.clearTimer);
                        if(o.mouseDownElem != null) {
                                removeEvent(o.mouseDownElem, "mouseout",
o.clearTimer);
                        };
                        o.mouseDownElem = null;
                };

                var o = this;
```

```
                if(this.staticPos) {
                        this.create();
                } else {
                        this.createButton();
                };

                this.setDateFromInput();

                (function() {
                        var elem = o.getElem();
                        if(elem && elem.tagName && elem.tagName.search(/select|
input/i) != -1) {
                                addEvent(elem, "change", o.changeHandler);
                                if(this.splitDate) {
                                        addEvent(document.getElementById(o.id +
splitAppend[1]), "change", o.changeHandler);
                                        addEvent(document.getElementById(o.id +
splitAppend[0]), "change", o.changeHandler);
                                };
                        };

                        if(!elem || elem.disabled == true) {
                               o.disableDatePicker();
                        };
                })();

                // We have fully created the datepicker...
                this.fullCreate = true;
        };
        datePicker.prototype.addButtonEvents = function(but) {
                but.onkeydown = but.onclick = function(e) {
                        e = e || window.event;

                        var inpId     = this.id.replace('fd-but-',''),
                            dpVisible = isVisible(inpId),
                            autoFocus = false;

                        if(e.type == "keydown") {
                                var kc = e.keyCode != null ? e.keyCode :
e.charCode;
                                if(kc != 13) return true;
                                if(dpVisible) {
                                        this.className =
this.className.replace("dp-button-active", "");
                                        hideAll();
                                        return false;
                                };
                                autoFocus = true;
                        };

                        this.className = this.className.replace("dp-button-
active", "");
```

```
                        if(!dpVisible) {
                                this.className += " dp-button-active";
                                hideAll(inpId);
                                showDatePicker(inpId, autoFocus);
                        } else {
                                hideAll();
                        };

                        return false;
                };

                if(!buttonTabIndex) {
                        but.setAttribute(!/*@cc_on!@*/false ? "tabIndex" :
"tabindex", "-1");
                        but.tabIndex = -1;
                        but.onkeydown = null;
                } else {
                        but.setAttribute(!/*@cc_on!@*/false ? "tabIndex" :
"tabindex", "0");
                        but.tabIndex = 0;
                };
        };

        datePicker.prototype.createButton = function() {

                if(this.staticPos || document.getElementById("fd-but-" + this.id))
{ return; };

                var inp         = this.getElem(),
                    span        = document.createElement('span'),
                    but         = document.createElement('a');

                but.href        = "#" + this.id;
                but.className   = "date-picker-control";
                but.title       = getTitleTranslation(5);
                but.id          = "fd-but-" + this.id;

                span.appendChild(document.createTextNode(nbsp));
                but.appendChild(span);

                span = document.createElement('span');
                span.className = "fd-screen-reader";
                span.appendChild(document.createTextNode(but.title));
                but.appendChild(span);

                // Set the ARIA role to be "button"
                setARIARole(but, "button");

                // Set a "haspopup" ARIA property - should this not be a list if
ID's????
                setARIAProperty(but, "haspopup", true);

                if(this.buttonWrapper &&
document.getElementById(this.buttonWrapper)) {
```

```
document.getElementById(this.buttonWrapper).appendChild(but);
                } else if(inp.nextSibling) {
                        inp.parentNode.insertBefore(but, inp.nextSibling);
                } else {
                        inp.parentNode.appendChild(but);
                };

                this.addButtonEvents(but);

                but = null;
        };
        datePicker.prototype.setRangeLow = function(range) {
                this.low = (String(range).search(/^(\d\d\d\d)(0[1-9]|1[012])(0[1-
9]|[12][0-9]|3[01])$/) == -1) ? false : range;
                this.checkSelectedDate();
                if(this.created) { this.updateTable(); };
        };
        datePicker.prototype.setRangeHigh = function(range) {
                this.high = (String(range).search(/^(\d\d\d\d)(0[1-9]|1[012])(0[1-
9]|[12][0-9]|3[01])$/) == -1) ? false : range;
                this.checkSelectedDate();
                if(this.created) { this.updateTable(); };
        };
        datePicker.prototype.setDisabledDays = function(dayArray) {
                this.disabledDays = dayArray;
                this.checkSelectedDate();
                if(this.created) { this.updateTable(); };
        };
        datePicker.prototype.setDisabledDates = function(dateArray) {
                this.disabledDates = {};
                this.addDisabledDates(dateArray);
        };
        datePicker.prototype.addDisabledDates = function(dateArray) {
                var disabledDateObj = {};
                if(typeof dateArray !== "object") dateArray = [dateArray];
                for(var i = dateArray.length; i-- ;) {
                        if(dateArray[i].match(/^(\d\d\d\d|\*\*\*\*)(0[1-9]|
1[012]|\*\*)(0[1-9]|[12][0-9]|3[01])$/) != -1) {
                                this.disabledDates[dateArray[i]] = 1;
                        };
                };
                this.checkSelectedDate();
                if(this.created) { this.updateTable(); };
        };
        datePicker.prototype.checkSelectedDate = function() {
                if(!this.dateSet) return;
                if(!this.canDateBeSelected(this.dateSet)) {
                        this.dateSet = null;
                };
        };
        datePicker.prototype.addFocusEvents = function() {
                if(this.kbEventsAdded || this.noFocus) {
                        return;
```

```
                };

                this.div.className = this.div.className.replace(/datepicker-
focus/, "") + " datepicker-focus";
                addEvent(document, "keypress", this.onkeydown);
                addEvent(document, "mousedown", this.onmousedown);

                /*@cc_on
                @if(@_win32)
                removeEvent(document, "keypress", this.onkeydown);
                addEvent(document, "keydown", this.onkeydown);
                @end
                @*/
                if(window.devicePixelRatio) {
                        removeEvent(document, "keypress", this.onkeydown);
                        addEvent(document, "keydown", this.onkeydown);
                };
                this.noFocus = false;
                this.kbEventsAdded = true;
        };
        datePicker.prototype.removeFocusEvents = function() {
                if(!this.kbEventsAdded) { return; };
                this.div.className = this.div.className.replace(/datepicker-
focus/, "");
                removeEvent(document, "keypress",  this.onkeydown);
                removeEvent(document, "keydown",   this.onkeydown);
                removeEvent(document, "mousedown", this.onmousedown);
                this.noFocus       = true;
                this.kbEventsAdded = false;
        };
        datePicker.prototype.stopTimer = function() {
                this.timerSet = false;
                window.clearTimeout(this.timer);
        };
        datePicker.prototype.setOpacity = function(op) {
                this.div.style.opacity = op/100;
                this.div.style.filter = 'alpha(opacity=' + op + ')';
                this.opacity = op;
        };
        datePicker.prototype.getElem = function() {
                return document.getElementById(this.id.replace(/^fd-/, '')) ||
false;
        };
        datePicker.prototype.getEnabledDates = function(y, m) {
                m = pad(m);

                var obj = {},
                    lower  = this.firstDateShown,
                    upper  = this.lastDateShown,
                    dt1, dt2, rngLower, rngUpper;

                if(!upper || !lower) {
                        lower = this.firstDateShown = y + pad(m) + "01";
                        upper = this.lastDateShown  = y + pad(m) +
```

```
pad(daysInMonth(m, y));
                };

                for(dt in this.enabledDates) {
                        dt1 = dt.replace(/^(\*\*\*\*)/, y).replace(/^(\d\d\d\d)
(\*\*)/, "$1"+m);
                        dt2 = this.enabledDates[dt];

                        if(dt2 == 1) {
                                obj[dt1] = 1;
                                continue;
                        };

                        // Range
                        //if(Number(dt1.substr(0,6)) ==
+String(this.firstDateShown).substr(0,6) && Number(dt2.substr(0,6)) ==
+String(this.lastDateShown).substr(0,6)) {
                                // Same month
                                if(Number(dt1.substr(0,6)) ==
Number(dt2.substr(0,6))) {
                                        for(var i = dt1; i <= dt2; i++) {
                                                obj[i] = 1;
                                        };
                                        continue;
                                };

                                // Different months but we only want this month
                                rngLower = Number(dt1.substr(0,6)) ==
+String(this.firstDateShown).substr(0,6) ? dt1 : lower;
                                rngUpper = Number(dt2.substr(0,6)) ==
+String(this.lastDateShown).substr(0,6) ? dt2 : upper;
                                for(var i = +rngLower; i <= +rngUpper; i++) {
                                        obj[i] = 1;
                                };
                        //};
                };
                return obj;
        };

        datePicker.prototype.getDisabledDates = function(y, m) {
                m = pad(m);

                var obj = {},
                    lower  = this.firstDateShown,
                    upper  = this.lastDateShown,
                    dt1, dt2, rngLower, rngUpper;

                if(!upper || !lower) {
                        lower = this.firstDateShown = y + pad(m) + "01";
                        upper = this.lastDateShown  = y + pad(m) +
pad(daysInMonth(m, y));
                };

                for(var dt in this.disabledDates) {
```

```
                        dt1 = dt.replace(/^(\*\*\*\*)/, y).replace(/^(\d\d\d\d)
(\*\*)/, "$1"+m);
                        dt2 = this.disabledDates[dt];

                        if(dt2 == 1) {
                                if(+lower <= +dt1 && +upper >= +dt1) {
                                        obj[dt1] = 1;
                                };
                                continue;
                        };

                        // Range of disabled dates
                        if(Number(dt1.substr(0,6)) <=
+String(this.firstDateShown).substr(0,6) && Number(dt2.substr(0,6)) >=
+String(this.lastDateShown).substr(0,6)) {
                                // Same month
                                if(Number(dt1.substr(0,6)) ==
Number(dt2.substr(0,6))) {
                                        for(var i = dt1; i <= dt2; i++) {
                                                obj[i] = 1;
                                        };
                                        continue;
                                };

                                // Different months but we only want this month
                                rngLower = Number(dt1.substr(0,6)) ==
+String(this.firstDateShown).substr(0,6) ? dt1 : lower;
                                rngUpper = Number(dt2.substr(0,6)) ==
+String(this.lastDateShown).substr(0,6) ? dt2 : upper;
                                for(var i = +rngLower; i <= +rngUpper; i++) {
                                        obj[i] = 1;
                                };
                        };
                };

                for(dt in this.enabledDates) {
                        dt1 = dt.replace(/^(\*\*\*\*)/, y).replace(/^(\d\d\d\d)
(\*\*)/, "$1"+m);
                        dt2 = this.enabledDates[dt];

                        if(dt2 == 1) {
                                if(dt1 in obj) {
                                        obj[dt1] = null;
                                        delete obj[dt1];
                                };
                                continue;
                        };

                        // Range
                        if(Number(dt1.substr(0,6)) <=
+String(this.firstDateShown).substr(0,6) && Number(dt2.substr(0,6)) >=
+String(this.lastDateShown).substr(0,6)) {
                                // Same month
                                if(Number(dt1.substr(0,6)) ==
```

```
Number(dt2.substr(0,6))) {
                                        for(var i = dt1; i <= dt2; i++) {
                                                if(i in obj) {
                                                        obj[i] = null;
                                                        delete obj[i];
                                                };
                                        };
                                        continue;
                                };

                                // Different months but we only want this month
                                rngLower = Number(dt1.substr(0,6)) ==
+String(this.firstDateShown).substr(0,6) ? dt1 : lower;
                                rngUpper = Number(dt2.substr(0,6)) ==
+String(this.lastDateShown).substr(0,6) ? dt2 : upper;
                                for(var i = +rngLower; i <= +rngUpper; i++) {
                                        if(i in obj) {
                                                obj[i] = null;
                                                delete obj[i];
                                        };
                                };
                        };
                };
                return obj;
        };
        datePicker.prototype.truePosition = function(element) {
                var pos = this.cumulativeOffset(element);
                if(window.opera) { return pos; };
                var iebody      = (document.compatMode && document.compatMode !=
"BackCompat")? document.documentElement : document.body,
                    dsocleft    = document.all ? iebody.scrollLeft :
window.pageXOffset,
                    dsoctop     = document.all ? iebody.scrollTop  :
window.pageYOffset,
                    posReal     = this.realOffset(element);
                return [pos[0] - posReal[0] + dsocleft, pos[1] - posReal[1] +
dsoctop];
        };
        datePicker.prototype.realOffset = function(element) {
                var t = 0, l = 0;
                do {
                        t += element.scrollTop  || 0;
                        l += element.scrollLeft || 0;
                        element = element.parentNode;
                } while(element);
                return [l, t];
        };
        datePicker.prototype.cumulativeOffset = function(element) {
                var t = 0, l = 0;
                do {
                        t += element.offsetTop  || 0;
                        l += element.offsetLeft || 0;
                        element = element.offsetParent;
                } while(element);
```

```
                return [l, t];
        };
        datePicker.prototype.equaliseDates = function() {
                var clearDayFound = false, tmpDate;
                for(var i = this.low; i <= this.high; i++) {
                        tmpDate = String(i);
                        if(!this.disabledDays[new Date(tmpDate.substr(0,4),
tmpDate.substr(6,2), tmpDate.substr(4,2)).getDay() - 1]) {
                                clearDayFound = true;
                                break;
                        };
                };
                if(!clearDayFound) { this.disabledDays = [0,0,0,0,0,0,0]__};
        };
        datePicker.prototype.outOfRange = function(tmpDate) {
                if(!this.low && !this.high) { return false; };

                var level = false;
                if(!tmpDate) {
                        level   = true;
                        tmpDate = this.date;
                };

                var d  = pad(tmpDate.getDate()),
                    m  = pad(tmpDate.getMonth() + 1),
                    y  = tmpDate.getFullYear(),
                    dt = String(y)+String(m)+String(d);

                if(this.low && +dt < +this.low) {
                        if(!level) { return true; };
                        this.date = new Date(this.low.substr(0,4),
this.low.substr(4,2)-1, this.low.substr(6,2), 5, 0, 0);
                        return false;
                };
                if(this.high && +dt > +this.high) {
                        if(!level) { return true; };
                        this.date = new Date(this.high.substr(0,4),
this.high.substr(4,2)-1, this.high.substr(6,2), 5, 0, 0);
                };
                return false;
        };
        datePicker.prototype.canDateBeSelected = function(tmpDate) {
                if(!tmpDate) return false;

                var d  = pad(tmpDate.getDate()),
                    m  = pad(tmpDate.getMonth() + 1),
                    y  = tmpDate.getFullYear(),
                    dt = String(y)+String(m)+String(d),
                    dd = this.getDisabledDates(+y, +m),
                    de = this.getEnabledDates(+y, +m),
                    wd = printFormattedDate(tmpDate, "N");

                if((this.low && +dt < +this.low) || (this.high && +dt >
+this.high) || (dt in dd) || (this.disabledDays[wd-1] && !(dt in de))) {
```

```
                        return false;
                };

                return true;
        };
        datePicker.prototype.updateStatus = function(msg) {
                while(this.statusBar.firstChild) {
this.statusBar.removeChild(this.statusBar.firstChild); };

                if(msg && this.statusFormat.search(/-S|S-/) != -1 && msg.search(/
([0-9]{1,2})(st|nd|rd|th)/) != -1) {
                        msg = msg.replace(/([0-9]{1,2})(st|nd|rd|th)/,
"$1<sup>$2</sup>").split(/<sup>|<\/sup>/);
                        var dc = document.createDocumentFragment();
                        for(var i = 0, nd; nd = msg[i]; i++) {
                                if(/^(st|nd|rd|th)$/.test(nd)) {
                                        var sup = document.createElement("sup");

sup.appendChild(document.createTextNode(nd));
                                        dc.appendChild(sup);
                                } else {

dc.appendChild(document.createTextNode(nd));
                                };
                        };
                        this.statusBar.appendChild(dc);
                } else {
                        this.statusBar.appendChild(document.createTextNode(msg ?
msg : getTitleTranslation(9)));
                };
        };
        datePicker.prototype.setDateFromInput = function() {
                this.dateSet = null;

                var elem = this.getElem(),
                    upd  = false,
                    dt;

                if(!elem || elem.tagName.search(/select|input/i) == -1) return;

                if(!this.splitDate && elem.value.replace(/\s/g, "") !== "") {
                        var dynFormatMasks =
formatMasks.concat([this.format]).reverse();
                        for(var i = 0, fmt; fmt = dynFormatMasks[i]; i++) {
                                dt = parseDateString(elem.value, fmt);
                                if(dt) {
                                        upd = true;
                                        break;
                                };
                        };
                } else if(this.splitDate) {
                        var mmN  = document.getElementById(this.id +
splitAppend[1]),
                            ddN  = document.getElementById(this.id +
```

```
splitAppend[0]),
                                tm   = parseInt(mmN.tagName.toLowerCase() == "input"
? mmN.value  : mmN.options[mmN.selectedIndex || 0].value, 10),
                                td   = parseInt(ddN.tagName.toLowerCase() == "input"
? ddN.value  : ddN.options[ddN.selectedIndex || 0].value, 10),
                                ty   = parseInt(elem.tagName.toLowerCase() == "input"
? elem.value : elem.options[elem.selectedIndex || 0].value, 10);

                        if(!(/\d\d\d\d/.test(ty)) || !(/^(0?[1-9]|1[012])
$/.test(tm)) || !(/^(0?[1-9]|[12][0-9]|3[01])$/.test(td))) {
                                dt = false;
                        } else {
                                if(+td > daysInMonth(+tm - 1, +ty)) {
                                        upd = true;
                                        td  = daysInMonth(+tm - 1, +ty);
                                        dt  = new Date(ty,tm-1,td);
                                } else {
                                        dt = new Date(ty,tm-1,td);
                                };
                        };
                };

                if(!dt || isNaN(dt)) {
                        this.date = this.cursorDate ? new
Date(+this.cursorDate.substr(0,4), +this.cursorDate.substr(4,2) - 1,
+this.cursorDate.substr(6,2)) : new Date();
                        this.date.setHours(5);
                        this.outOfRange();
                        return;
                };

                dt.setHours(5);
                this.date = new Date(dt);
                this.outOfRange();

                if(dt.getTime() == this.date.getTime() &&
this.canDateBeSelected(this.date)) {
                        this.dateSet = new Date(this.date);
                };

                if(upd) { this.returnFormattedDate(true); };
        };
        datePicker.prototype.setSelectIndex = function(elem, indx) {
                for(var opt = elem.options.length-1; opt >= 0; opt--) {
                        if(elem.options[opt].value == +indx) {
                                elem.selectedIndex = opt;
                                return;
                        };
                };
        };
        datePicker.prototype.returnFormattedDate = function(noFocus) {
                var elem = this.getElem();
                if(!elem || this.dateSet == null) return;
```

```
                noFocus = !!noFocus;

                var d                   = pad(this.date.getDate()),
                    m                   = pad(this.date.getMonth() + 1),
                    yyyy                = this.date.getFullYear(),
                    disabledDates       = this.getDisabledDates(+yyyy, +m),
                    enabledDates        = this.getEnabledDates(+yyyy, +m),
                    weekDay             = (this.date.getDay() + 6) % 7,
                    dt                  = String(yyyy) + String(m) + String(d);

                if(!(this.disabledDays[weekDay] && !(dt in enabledDates)) || !
(String(yyyy)+m+d in this.disabledDates)) {
                        if(this.splitDate) {
                                var ddE =
document.getElementById(this.id+splitAppend[0]),
                                    mmE =
document.getElementById(this.id+splitAppend[1]),
                                    tY = elem.value, tM = mmE.value, tD =
ddE.value;

                                if(+tY == +yyyy && +tM == +m && +tD == +d) {
return; };

                                if(ddE.tagName.toLowerCase() == "input") {
ddE.value = d; }
                                else { this.setSelectIndex(ddE, d); };
                                if(mmE.tagName.toLowerCase() == "input") {
mmE.value = m; }
                                else { this.setSelectIndex(mmE, m); };
                                if(elem.tagName.toLowerCase() == "input")
elem.value = yyyy;
                                else { this.setSelectIndex(elem, yyyy); };
                        } else if(elem.tagName.toLowerCase() == "input") {
                                var oldVal = elem.value,
                                    newVal = printFormattedDate(this.date,
this.format, returnLocaleDate);

                                if(oldVal == newVal) { return; };
                                elem.value = newVal;
                        };

                        if(this.staticPos) {
                                this.noFocus = true;
                                this.updateTable();
                                this.noFocus = false;
                        };

                        if(this.fullCreate) {
                                if(elem.type && elem.type != "hidden" && !noFocus)
{ elem.focus(); };
                                this.callback("dateselect", { "id":this.id,
"date":this.dateSet, "dd":d, "mm":m, "yyyy":yyyy });
                        };
                };
```

```
	};
	datePicker.prototype.disableDatePicker = function() {
		if(this.disabled) return;

		if(this.staticPos) {
			this.removeFocusEvents();
			this.noFocus = true;
			this.div.className = this.div.className.replace(/dp-
disabled/, "") + " dp-disabled";
			this.table.onmouseover = this.table.onclick =
this.table.onmouseout = this.table.onmousedown = null;
			removeEvent(document, "mousedown", this.onmousedown);
			removeEvent(document, "mouseup",   this.clearTimer);
		} else {
			if(this.visible) this.hide();
			var but = document.getElementById("fd-but-" + this.id);
			if(but) {
				but.className = but.className.replace(/dp-
disabled/, "") + " dp-disabled";
				// Set a "disabled" ARIA state
				setARIAProperty(but, "disabled", true);
				but.onkeydown = but.onclick = function() { return
false; };

				but.setAttribute(!/*@cc_on!@*/false ? "tabIndex" :
"tabindex", "-1");
				but.tabIndex = -1;
			};
		};

		clearTimeout(this.timer);
		this.disabled = true;
	};
	datePicker.prototype.enableDatePicker = function() {
		if(!this.disabled) return;

		if(this.staticPos) {
			this.removeOldFocus();
			this.noFocus = true;
			this.addSpans = true;
			this.updateTable();
			this.div.className = this.div.className.replace(/dp-
disabled/, "");
			this.disabled = false;
			this.table.onmouseover = this.onmouseover;
			this.table.onmouseout  = this.onmouseout;
			this.table.onclick     = this.onclick;
			this.table.onmousedown = this.onmousedown;
		} else {
			var but = document.getElementById("fd-but-" + this.id);
			if(but) {
				but.className = but.className.replace(/dp-
disabled/, "");
				// Reset the "disabled" ARIA state
				setARIAProperty(but, "disabled", false);
```

```
                                this.addButtonEvents(but);
                        };
                };

                this.disabled = false;
        };
        datePicker.prototype.disableTodayButton = function() {
                var today = new Date();
                this.butToday.className = this.butToday.className.replace("fd-
disabled", "");
                if(this.outOfRange(today) || (this.date.getDate() ==
today.getDate() && this.date.getMonth() == today.getMonth() &&
this.date.getFullYear() == today.getFullYear())) {
                        this.butToday.className += " fd-disabled";
                };
        };
        datePicker.prototype.updateTableHeaders = function() {
                var colspanTotal = this.showWeeks ? 8 : 7,
                    colOffset    = this.showWeeks ? 1 : 0,
                    d, but;

                for(var col = colOffset; col < colspanTotal; col++ ) {
                        d = (this.firstDayOfWeek + (col - colOffset)) % 7;
                        this.ths[col].title = getDayTranslation(d, false);

                        if(col > colOffset) {
                                but = this.ths[col].getElementsByTagName("span")
[0];
                                while(but.firstChild) {
but.removeChild(but.firstChild); };

but.appendChild(document.createTextNode(getDayTranslation(d, true)));
                                but.title = this.ths[col].title;
                                but.className = but.className.replace(/day-([0-
6])/, "") + " day-" + d;
                                but = null;
                        } else {
                                while(this.ths[col].firstChild) {
this.ths[col].removeChild(this.ths[col].firstChild); };

this.ths[col].appendChild(document.createTextNode(getDayTranslation(d, true)));
                        };

                        this.ths[col].className =
this.ths[col].className.replace(/date-picker-highlight/g, "");
                        if(this.highlightDays[d]) {
                                this.ths[col].className += " date-picker-
highlight";
                        };
                };

                if(this.created) { this.updateTable(); }
        };
        datePicker.prototype.callback = function(type, args) {
```

```
                if(!type || !(type in this.callbacks)) return false;

                var ret = false;
                for(var func = 0; func < this.callbacks[type].length; func++) {
                        ret = this.callbacks[type][func](args || this.id);
                        if(!ret) return false;
                };
                return ret;
        };
        datePicker.prototype.showHideButtons = function(tmpDate) {
                var tdm = tmpDate.getMonth(),
                    tdy = tmpDate.getFullYear();

                this.butPrevYear.className =
this.butPrevYear.className.replace("fd-disabled", "");
                if(this.outOfRange(new Date((tdy - 1), tdm, daysInMonth(+tdm, tdy-
1)))) {
                        this.butPrevYear.className += " fd-disabled";
                        if(this.yearInc == -1) this.stopTimer();
                };

                this.butPrevMonth.className =
this.butPrevMonth.className.replace("fd-disabled", "");
                if(this.outOfRange(new Date(tdy, (+tdm - 1), daysInMonth(+tdm-1,
tdy)))) {
                        this.butPrevMonth.className += " fd-disabled";
                        if(this.monthInc == -1) this.stopTimer();
                };

                this.butNextYear.className =
this.butNextYear.className.replace("fd-disabled", "");
                if(this.outOfRange(new Date((tdy + 1), +tdm, 1))) {
                        this.butNextYear.className += " fd-disabled";
                        if(this.yearInc == 1) this.stopTimer();
                };

                this.butNextMonth.className =
this.butNextMonth.className.replace("fd-disabled", "");
                if(this.outOfRange(new Date(tdy, +tdm + 1, 1))) {
                        this.butNextMonth.className += " fd-disabled";
                        if(this.monthInc == 1) this.stopTimer();
                };
        };

        var grepRangeLimits = function(sel) {
                var range = [];
                for(var i = 0; i < sel.options.length; i++) {
                        if(sel.options[i].value.search(/^\d\d\d\d$/) == -1) {
continue; };
                        if(!range[0] || Number(sel.options[i].value) < range[0]) {
range[0] = Number(sel.options[i].value); };
                        if(!range[1] || Number(sel.options[i].value) > range[1]) {
range[1] = Number(sel.options[i].value); };
                };
```

```
                return range;
        };
        var joinNodeLists = function() {
                if(!arguments.length) { return []; }
                var nodeList = [];
                for (var i = 0; i < arguments.length; i++) {
                        for (var j = 0, item; item = arguments[i][j]; j++) {
                                nodeList[nodeList.length] = item;
                        };
                };
                return nodeList;
        };
        var cleanUp = function() {
                var dp;
                for(dp in datePickers) {
                        if(!document.getElementById(datePickers[dp].id)) {
                                datePickers[dp].destroy();
                                datePickers[dp] = null;
                                delete datePickers[dp];
                        };
                };
        };
        var hideAll = function(exception) {
                var dp;
                for(dp in datePickers) {
                        if(!datePickers[dp].created || (exception && exception ==
datePickers[dp].id)) continue;
                        datePickers[dp].hide();
                };
        };
        var hideDatePicker = function(inpID) {
                if(inpID in datePickers) {
                        if(!datePickers[inpID].created ||
datePickers[inpID].staticPos) return;
                        datePickers[inpID].hide();
                };
        };
        var showDatePicker = function(inpID, autoFocus) {
                if(!(inpID in datePickers)) return false;
                datePickers[inpID].show(autoFocus);
                return true;
        };
        var destroy = function() {
                for(dp in datePickers) {
                        datePickers[dp].destroy();
                        datePickers[dp] = null;
                        delete datePickers[dp];
                };
                datePickers = null;
                removeEvent(window, 'load',   datePickerController.create);
                removeEvent(window, 'unload', datePickerController.destroy);
        };
        var destroySingleDatePicker = function(id) {
                if(id && (id in datePickers)) {
```

```
                        datePickers[id].destroy();
                        datePickers[id] = null;
                        delete datePickers[id];
                };
        };
        var getTitleTranslation = function(num, replacements) {
                replacements = replacements || [];
                if(localeImport.titles.length > num) {
                        var txt = localeImport.titles[num];
                        if(replacements && replacements.length) {
                                for(var i = 0; i < replacements.length; i++) {
                                        txt = txt.replace("[[%" + i + "%]]",
replacements[i]);
                                };
                        };
                        return txt.replace(/[[%(\d)%]]/g,"");
                };
                return "";
        };
        var getDayTranslation = function(day, abbreviation) {
                var titles = localeImport[abbreviation ? "dayAbbrs" : "fullDays"];
                return titles.length && titles.length > day ? titles[day] : "";
        };
        var getMonthTranslation = function(month, abbreviation) {
                var titles = localeImport[abbreviation ? "monthAbbrs" :
"fullMonths"];
                return titles.length && titles.length > month ? titles[month] :
"";
        };
        var daysInMonth = function(nMonth, nYear) {
                nMonth = (nMonth + 12) % 12;
                return (((0 == (nYear%4)) && ((0 != (nYear%100)) || (0 == (nYear
%400)))) && nMonth == 1) ? 29: [31,28,31,30,31,30,31,31,30,31,30,31][nMonth];
        };
        var getWeeksInYear = function(Y) {
                if(Y in weeksInYearCache) {
                        return weeksInYearCache[Y];
                };
                var X1, X2, NW;
                with (X1 = new Date(Y, 0, 4)) {
                        setDate(getDate() - (6 + getDay()) % 7);
                };
                with (X2 = new Date(Y, 11, 28)) {
                        setDate(getDate() + (7 - getDay()) % 7);
                };
                weeksInYearCache[Y] = Math.round((X2 - X1) / 604800000);
                return weeksInYearCache[Y];
        };
        var parseRangeFromString = function(str) {
                if(!str) return "";

                var low = str.search(/^range-low-/) != -1;
                str = str.replace(/range-(low|high)-/, "");
```

```
                if(str.search(/^(\d\d\d\d)(0[1-9]|1[012])(0[1-9]|[12][0-9]|3[01])
$/) != -1) { return str; };

                var tmpDate = new Date();

                if(str.search(/^today$/) != -1) { return tmpDate.getFullYear() +
pad(tmpDate.getMonth() + 1) + pad(tmpDate.getDate()); };

                var regExp = /^(\d)-(day|week|month|year)$/;

                if(str.search(regExp) != -1) {
                        var parts       = str.match(regExp),
                            acc         = { day:0,week:0,month:0,year:0 };

                        acc[parts[2]]   = low ? -(+parts[1]) : +parts[1];
                        tmpDate.setFullYear(tmpDate.getFullYear() + +acc.year);
                        tmpDate.setMonth(tmpDate.getMonth() + +acc.month);
                        tmpDate.setDate(tmpDate.getDate() + +acc.day + (7 *
+acc.week));
                        return !tmpDate || isNaN(tmpDate) ? "" :
tmpDate.getFullYear() + pad(tmpDate.getMonth() + 1) + pad(tmpDate.getDate());
                };

                return "";
        };
        var getWeekNumber = function(y,m,d) {
                var d = new Date(y, m, d, 0, 0, 0);
                var DoW = d.getDay();
                d.setDate(d.getDate() - (DoW + 6) % 7 + 3); // Nearest Thu
                var ms = d.valueOf(); // GMT
                d.setMonth(0);
                d.setDate(4); // Thu in Week 1
                return Math.round((ms - d.valueOf()) / (7 * 864e5)) + 1;
        };
        var printFormattedDate = function(date, fmt, useImportedLocale) {
                if(!date || isNaN(date)) { return ""; };

                var parts = fmt.split("-"),
                      str = [],
                        d = date.getDate(),
                        D = date.getDay(),
                        m = date.getMonth(),
                        y = date.getFullYear(),
                    flags = {
                                    "sp":" ",
                                    "dt":".",
                                    "sl":"/",
                                    "ds":"-",
                                    "cc":",",
                                    "d":pad(d),
                                    "D":useImportedLocale ? localeImport.dayAbbrs[D ==
0 ? 6 : D - 1] : localeDefaults.dayAbbrs[D == 0 ? 6 : D - 1],
                                    "l":useImportedLocale ? localeImport.fullDays[D ==
0 ? 6 : D - 1] : localeDefaults.fullDays[D == 0 ? 6 : D - 1],
```

```
                                "j":d,
                                "N":D == 0 ? 7 : D,
                                "w":D,
                                "W":getWeekNumber(date),
                                "M":useImportedLocale ? localeImport.monthAbbrs[m]
: localeDefaults.monthAbbrs[m],
                                "F":useImportedLocale ? localeImport.fullMonths[m]
: localeDefaults.fullMonths[m],
                                "m":pad(m + 1),
                                "n":m + 1,
                                "t":daysInMonth(m + 1, y),
                                "Y":y,
                                "o":y,
                                "y":String(y).substr(2,2),
                                "S":["th", "st", "nd", "rd"][d % 10 > 3 ? 0 : (d %
100 - d % 10 != 10) * d % 10]
                            };

                for(var pt = 0, part; part = parts[pt]; pt++) {
                        str.push(!(part in flags) ? "" : flags[part]);
                };

                return str.join("");
        };
        var parseDateString = function(str, fmt) {
                var d     = false,
                    m     = false,
                    y     = false,
                    now   = new Date(),
                    parts = fmt.replace(/-sp(-sp)+/g, "-sp").split("-"),
                    divds = { "dt":".","sl":"/","ds":"-","cc":"," };

                loopLabel:
                for(var pt = 0, part; part = parts[pt]; pt++) {
                        if(str.length == 0) { return false; };

                        switch(part) {
                                // Dividers
                                case "sp": // Space " "
                                                if(str.charAt(0).search(/\s/) !=
-1) {
                                                        // Be easy on multiple
spaces...

while(str.charAt(0).search(/\s/) != -1) { str = str.substr(1); };
                                                        break;
                                                } else return "";
                                case "dt":
                                case "sl":
                                case "ds":
                                case "cc":
                                                if(str.charAt(0) == divds[part]) {
                                                        str = str.substr(1);
                                                        break;
```

```
                                        } else return "";
                                // DAY
                                case "d": // Day of the month, 2 digits with
leading zeros (01 - 31)
                                case "j": // Day of the month without leading
zeros (1 - 31)
                                        // Accept both when parsing
                                        if(str.search(/^(3[01]|[12][0-9]|
0?[1-9])/) != -1) {
                                                d = +str.match(/^(3[01]|
[12][0-9]|0?[1-9])/)[0];
                                                str =
str.substr(str.match(/^(3[01]|[12][0-9]|0?[1-9])/)[0].length);
                                                break;
                                        } else return "";

                                case "D": // A textual representation of a day,
three letters (Mon - Sun)
                                case "l": // A full textual representation of the
day of the week (Monday - Sunday)
                                        l = part == "D" ?
localeDefaults.dayAbbrs : localeDefaults.fullDays;
                                        for(var i = 0; i < 7; i++) {
                                                if(new RegExp("^" + l[i],
"i").test(str)) {
                                                        str =
str.substr(l[i].length);
                                                        continue
loopLabel;
                                                };
                                        };
                                        return "";
                                case "N": // ISO-8601 numeric representation of
the day of the week (added in PHP 5.1.0) 1 (for Monday) through 7 (for Sunday)
                                case "w": // Numeric representation of the day of
the week 0 (for Sunday) through 6 (for Saturday)
                                        if(str.search(part == "N" ? /^([1-
7])/ : /^([0-6])/) != -1) {
                                                str = str.substr(1);
                                                break;
                                        } else return "";
                                case "S": // English ordinal suffix for the day of
the month, 2 characters: st, nd, rd or th
                                        if(str.search(/^(st|nd|rd|th)/i) !
= -1) {
                                                str = str.substr(2);
                                                break;
                                        } else return "";
                                case "z": // The day of the year (starting from
0): 0 - 365
                                        if(str.search(/^([0-9]|[1-9][0-9]|
[12][0-9]{2}|3[0-5][0-9]|36[0-5])/) != -1) {
                                                str =
str.substr(str.match(/^([0-9]|[1-9][0-9]|[12][0-9]{2}|3[0-5][0-9]|36[0-5])/)
```

```
[0].length);
                                                        break;
                                                } else return "";
                                        // WEEK
                                        case "W": // ISO-8601 week number of year, weeks
starting on Monday (added in PHP 4.1.0): 1 - 53
                                                if(str.search(/^([1-9]|[1234[0-9]|
5[0-3])/) != -1) {
                                                        str =
str.substr(str.match(/^([1-9]|[1234[0-9]|5[0-3])/)[0].length);
                                                        break;
                                                } else return "";
                                        // MONTH
                                        case "M": // A short textual representation of a
month, three letters
                                        case "F": // A full textual representation of a
month, such as January or March
                                                l =
localeDefaults.fullMonths.concat(localeDefaults.monthAbbrs); // :
localeDefaults.fullMonths;
                                                for(var i = 0; i < 24; i++) {
                                                        if(str.search(new
RegExp("^" + l[i],"i")) != -1) {
                                                                str =
str.substr(l[i].length);
                                                                m = ((i + 12) %
12);
                                                                continue
loopLabel;
                                                        };
                                                };
                                                return "";
                                        case "m": // Numeric representation of a month,
with leading zeros
                                        case "n": // Numeric representation of a month,
without leading zeros
                                                //l = part == "m" ? /^(0[1-9]|
1[012])/ : /^([1-9]|1[012])/;
                                                // Accept either when parsing
                                                l = /^(1[012]|0?[1-9])/;
                                                if(str.search(l) != -1) {
                                                        m = +str.match(l)[0] - 1;
                                                        str =
str.substr(str.match(l)[0].length);
                                                        break;
                                                } else return "";
                                        case "t": // Number of days in the given month: 28
through 31
                                                if(str.search(/2[89]|3[01]/) !=
-1) {
                                                        str = str.substr(2);
                                                        break;
                                                } else return "";
                                        // YEAR
```

```
                                case "Y": // A full numeric representation of a
year, 4 digits
                                case "o": // ISO-8601 year number. This has the
same value as Y
                                        if(str.search(/^(\d{4})/) != -1) {
                                                y = str.substr(0,4);
                                                str = str.substr(4);
                                                break;
                                        } else return "";
                                case "y": // A two digit representation of a year
- be easy on four figure dates though
                                        if(str.search(/^(\d{4})/) != -1) {
                                                y = str.substr(0,4);
                                                str = str.substr(4);
                                                break;
                                        } else if(str.search(/^(0[0-9]|[1-
9][0-9])/) != -1) {
                                                y = str.substr(0,2);
                                                y = +y < 50 ? '20' +
String(y) : '19' + String(y);
                                                str = str.substr(2);
                                                break;
                                        } else return "";
                                default:
                                        return "";
                        };
                };

                d = d === false ? now.getDate() : d;
                m = m === false ? now.getMonth() - 1 : m;
                y = y === false ? now.getFullYear() : y;

                var tmpDate = new Date(y,m,d);
                return isNaN(tmpDate) ? "" : tmpDate;
        };
        var repositionDatePickers = function(e) {
                for(dp in datePickers) {
                        if(!datePickers[dp].created || datePickers[dp].staticPos
|| (!datePickers[dp].staticPos && !datePickers[dp].dragDisabled)) continue;
                        datePickers[dp].reposition();
                };
        };
        var findLabelForElement = function(element) {
                var label;
                if(element.parentNode && element.parentNode.tagName.toLowerCase()
== "label") lebel = element.parentNode;
                else {
                        var labelList = document.getElementsByTagName('label');
                        // loop through label array attempting to match each 'for'
attribute to the id of the current element
                        for(var lbl = 0; lbl < labelList.length; lbl++) {
                                // Internet Explorer requires the htmlFor test
                                if((labelList[lbl]['htmlFor'] && labelList[lbl]
['htmlFor'] == element.id) || (labelList[lbl].getAttribute('for') == element.id))
```

```
{
                                        label = labelList[lbl];
                                        break;
                                };
                        };
                };

                if(label && !label.id) { label.id = element.id + "_label"; };

                return label;
        };
        var addDatePicker = function(options) {
                // Has the locale file loaded?
                if(typeof(fdLocale) == "object" && !localeImport) {
                        localeImport = {
                                titles          : fdLocale.titles,
                                fullMonths      : fdLocale.fullMonths,
                                monthAbbrs      : fdLocale.monthAbbrs,
                                fullDays        : fdLocale.fullDays,
                                dayAbbrs        : fdLocale.dayAbbrs,
                                firstDayOfWeek  : ("firstDayOfWeek" in fdLocale &&
String(fdLocale.firstDayOfWeek).search(/^([0-6]{1})$/) != -1) ?
fdLocale.firstDayOfWeek : 0,
                                imported        : true
                        };
                } else if(!localeImport) {
                        localeImport = localeDefaults;
                };

                if(!options.id || (options.id in datePickers)) { return; };

                var elem = document.getElementById(options.id);
                if(!elem || !elem.tagName || String(elem.tagName).search(/^(input|
select)$/i) == -1) { return; };

                if(!options.staticPos) {
                        options.hideInput       = false;
                } else {
                        options.noFadeEffect    = true;
                        options.dragDisabled    = true;
                };

                datePickers[options.id] = new datePicker(options);
        };
        var parseCallbacks = function(cbs) {
                if(cbs == null) { return {}; };
                var func,
                    type,
                    cbObj = {},
                    parts,
                    obj;
                for(var i = 0, fn; fn = cbs[i]; i++) {
                        type = fn.match(/(cb_(dateselect|redraw|create)_)([^\s|$]
+)/i)[1].replace(/^cb_/i, "").replace(/_$/, "");
```

```
                        fn   = fn.replace(/cb_(dateselect|redraw|create)_/i,
"").replace(/-/g, ".");

                        try {
                                if(fn.indexOf(".") != -1) {
                                        parts = fn.split('.');
                                        obj   = window;
                                        for (var x = 0, part; part =
obj[parts[x]]; x++) {
                                                if(part instanceof Function) {
                                                        (function() {
                                                                var method = part;
                                                                func = function
(data) { method.apply(obj, [data]) };
                                                        })();
                                                } else {
                                                obj = part;
                                                };
                                        };
                                } else {
                                        func = window[fn];
                                };

                                if(!(func instanceof Function)) continue;
                                if(!(type in cbObj)) { cbObj[type] = []; };
                                cbObj[type][cbObj[type].length] = func;
                        } catch (err) {};
                };
                return cbObj;
        };
        // Used by the button to dictate whether to open or close the datePicker
        var isVisible = function(id) {
                return (!id || !(id in datePickers)) ? false :
datePickers[id].visible;
        };
        var create = function(inp) {
                if(!(typeof document.createElement != "undefined" && typeof
document.documentElement != "undefined" && typeof
document.documentElement.offsetWidth == "number")) { return; };

                var formElements = (inp && inp.tagName) ? [inp] :
joinNodeLists(document.getElementsByTagName('input'),
document.getElementsByTagName('select')),
                    disableDays  = /disable-days-([1-7]){1,6}/g,
                    highlight    = /highlight-days-([1-7]{1,7})/,
                    rangeLow     = /range-low-((((\d\d\d\d)(0[1-9]|1[012])(0[1-9]|
[12][0-9]|3[01]))|((\d)-(day|week|month|year))|(today))/,
                    rangeHigh    = /range-high-((((\d\d\d\d)(0[1-9]|1[012])(0[1-9]|
[12][0-9]|3[01]))|((\d)-(day|week|month|year))|(today))/,
                    cursorDate   = /cursor-((\d\d\d\d)(0[1-9]|1[012])(0[1-9]|[12]
[0-9]|3[01]))/,
                    dateFormat   = /dateformat(-((sp|dt|sl|ds|cc)|([d|D|l|j|N|w|S|
W|M|F|m|n|t|Y|o|y|O|p])))+/,
                    statusFormat = /statusformat(-((sp|dt|sl|ds|cc)|([d|D|l|j|N|w|
```

```
S|W|M|F|m|n|t|Y|o|y|O|p]))+/,
                    disableDates = /disable((-(\d\d\d\d)(0[1-9]|1[012])(0[1-9]|
[12][0-9]|3[01])){2}|(-((\d\d\d\d)|(xxxx))((0[1-9]|1[012])|(xx))(0[1-9]|[12][0-9]|
3[01])))/g,
                    enableDates  = /enable((-(\d\d\d\d)(0[1-9]|1[012])(0[1-9]|[12]
[0-9]|3[01])){2}|(-((\d\d\d\d)|(xxxx))((0[1-9]|1[012])|(xx))(0[1-9]|[12][0-9]|
3[01])))/g,
                    callbacks    = /((cb_(dateselect|redraw|create)_)([^\s|$]
+))/ig,
                    positioned   = /display-inline-([^\s|$]+)/i,
                    bPositioned  = /button-([^\s|$]+)/i,
                    range,tmp,j,t,options,dts,parts;

                for(var i = 0, elem; elem = formElements[i]; i++) {
                        if(elem.className && (elem.className.search(dateFormat) !=
-1 || elem.className.search(/split-date/) != -1) && ((elem.tagName.toLowerCase()
== "input" && (elem.type == "text" || elem.type == "hidden")) ||
elem.tagName.toLowerCase() == "select")) {

                                if(elem.id && elem.id in datePickers) {
                                        if(!datePickers[elem.id].staticPos) {
datePickers[elem.id].createButton(); }
                                        else {
                                                if(!document.getElementById("fd-"
+ elem.id)) {

datePickers[elem.id].created = false;

datePickers[elem.id].create();
                                                } else if(inp) {
                                                        // Only do this if called
from an ajax update etc

datePickers[elem.id].setDateFromInput();

datePickers[elem.id].updateTable();
                                                };
                                        };
                                        continue;
                                };

                                if(!elem.id) { elem.id = "fdDatePickerInput-" +
uniqueId++; };

                                options = {
                                        id:elem.id,
                                        low:"",
                                        high:"",
                                        format:"d-sl-m-sl-Y",
                                        statusFormat:"",
                                        highlightDays:[0,0,0,0,0,1,1],
                                        disabledDays:[0,0,0,0,0,0,0],
                                        disabledDates:{},
                                        enabledDates:{},
```

```
                                        noFadeEffect:elem.className.search(/no-
animation/i) != -1,
                                        staticPos:elem.className.search(/display-
inline/i) != -1,
                                        hideInput:elem.className.search(/hide-
input/i) != -1,
                                        noToday:elem.className.search(/no-today-
button/i) != -1,
                                        showWeeks:elem.className.search(/show-
week/i) != -1,
                                        dragDisabled:nodrag ? true :
elem.className.search(/disable-drag/i) != -1,
                                        positioned:false,
                                        fillGrid:elem.className.search(/fill-
grid/i) != -1,
constrainSelection:elem.className.search(/fill-grid-no-select/i) != -1,
callbacks:parseCallbacks(elem.className.match(callbacks)),
                                        buttonWrapper:"",
                                        cursorDate:""
                                };

                                // The cursor start date
                                if(elem.className.search(cursorDate) != -1) {
                                        options.cursorDate =
elem.className.match(cursorDate)[1];
                                };

                                // Positioning of static dp's
                                if(options.staticPos &&
elem.className.search(positioned) != -1) {
                                        options.positioned =
elem.className.match(positioned)[1];
                                };

                                // Positioning of non-static dp's button
                                if(!options.staticPos &&
elem.className.search(bPositioned) != -1) {
                                        options.buttonWrapper =
elem.className.match(bPositioned)[1];
                                };

                                // Opacity of non-static datePickers
                                if(!options.staticPos) {
                                        options.finalOpacity =
elem.className.search(/opacity-([1-9]{1}[0-9]{1})/i) != -1 ?
elem.className.match(/opacity-([1-9]{1}[0-9]{1})/i)[1] : 90
                                };

                                // Dates to disable
                                dts = elem.className.match(disableDates);
                                if(dts) {
                                        for(t = 0; t < dts.length; t++) {
```

```
                                        parts = dts[t].replace(/xxxx/,
"****").replace(/xx/, "**").replace("disable-", "").split("-");
                                        options.disabledDates[parts[0]] =
(parts.length && parts.length == 2) ? parts[1] : 1;
                                }
                        };

                        // Dates to enable
                        dts = elem.className.match(enableDates);
                        if(dts) {
                                for(t = 0; t < dts.length; t++) {
                                        parts = dts[t].replace(/xxxx/,
"****").replace(/xx/, "**").replace("enable-", "").split("-");
                                        options.enabledDates[parts[0]] =
(parts.length && parts.length == 2) ? parts[1] : 1;
                                };
                        };

                        // Split the date into three parts ?
                        options.splitDate = (elem.className.search(/split-
date/) != -1 && document.getElementById(elem.id+splitAppend[0]) &&
document.getElementById(elem.id+splitAppend[1]) &&
document.getElementById(elem.id+splitAppend[0]).tagName.search(/input|select/i) !=
-1 && document.getElementById(elem.id+splitAppend[1]).tagName.search(/input|
select/i) != -1);

                        // Date format
                        if(!options.splitDate &&
elem.className.search(dateFormat) != -1) {
                                options.format =
elem.className.match(dateFormat)[0].replace('dateformat-','');
                        };

                        // Status bar date format
                        if(elem.className.search(statusFormat) != -1) {
                                options.statusFormat =
elem.className.match(statusFormat)[0].replace('statusformat-','');
                        };

                        // The days of the week to highlight
                        if(elem.className.search(highlight) != -1) {
                                tmp = elem.className.match(highlight)
[0].replace(/highlight-days-/, '');
                                options.highlightDays = [0,0,0,0,0,0,0];
                                for(j = 0; j < tmp.length; j++) {

options.highlightDays[tmp.charAt(j) - 1] = 1;
                                };
                        };

                        // The days of the week to disable
                        if(elem.className.search(disableDays) != -1) {
                                tmp = elem.className.match(disableDays)
[0].replace(/disable-days-/, '');
```

```
                                        options.disabledDays = [0,0,0,0,0,0,0];
                                        for(j = 0; j < tmp.length; j++) {
                                                options.disabledDays[tmp.charAt(j)
- 1] = 1;
                                        };
                                };

                                // The lower limit
                                if(elem.className.search(rangeLow) != -1) {
                                        options.low =
parseRangeFromString(elem.className.match(rangeLow)[0]);
                                };

                                // The higher limit
                                if(elem.className.search(rangeHigh) != -1) {
                                        options.high =
parseRangeFromString(elem.className.match(rangeHigh)[0]);
                                };

                                // Always round lower & higher limits if a
selectList involved
                                if(elem.tagName.search(/select/i) != -1) {
                                        range        = grepRangeLimits(elem);
                                        options.low  = options.low  ? range[0] +
String(options.low).substr(4,4)  : range[0] + "0101";
                                        options.high = options.high ? range[1] +
String(options.high).substr(4,4) : range[1] + "1231";
                                };

                                addDatePicker(options);
                        };
                };
        };

        addEvent(window, 'load',   create);
        addEvent(window, 'unload', destroy);
        addEvent(window, 'resize', repositionDatePickers);

        return {
                addEvent:               function(obj, type, fn) { return
addEvent(obj, type, fn); },
                removeEvent:            function(obj, type, fn) { return
removeEvent(obj, type, fn); },
                stopEvent:              function(e) { return stopEvent(e); },
                show:                   function(inpID) { return
showDatePicker(inpID, false); },
                hide:                   function(inpID) { return
hideDatePicker(inpID); },
                create:                 function(inp) { create(inp); },
                createDatePicker:       function(options) {
addDatePicker(options); },
                removeOnloadEvent:      function() { removeEvent(window, 'load',
create); },
                destroyDatePicker:      function(inpID) {
```

```
destroySingleDatePicker(inpID); },
                cleanUp:                function() { cleanUp(); },
                repositionDatePickers:  function() { repositionDatePickers(); },
                printFormattedDate:     function(dt, fmt, useImportedLocale) {
return printFormattedDate(dt, fmt, useImportedLocale); },
                setDateFromInput:       function(inpID) { if(!inpID || !(inpID in
datePickers) || !datePickers[inpID].created) return false;
datePickers[inpID].setDateFromInput(); },
                setRangeLow:            function(inpID, yyyymmdd) { if(!inpID || !
(inpID in datePickers)) return false; datePickers[inpID].setRangeLow(yyyymmdd); },
                setRangeHigh:           function(inpID, yyyymmdd) { if(!inpID || !
(inpID in datePickers)) return false; datePickers[inpID].setRangeHigh(yyyymmdd);
},
                parseDateString:        function(str, format) { return
parseDateString(str, format); },
                setGlobalVars:          function(json) { affectJSON(json); },
                dateValidForSelection:  function(inpID, dt) { if(!inpID || !(inpID
in datePickers)) return false; datePickers[inpID].canDateBeSelected(dt); },
                addDisabledDates:       function(inpID, dts) { if(!inpID || !
(inpID in datePickers)) return false; datePickers[inpID].addDisabledDates(dts); },
                setDisabledDates:       function(inpID, dts) { if(!inpID || !
(inpID in datePickers)) return false; datePickers[inpID].setDisabledDates(dts); },
                disable:                function(inpID) { if(!inpID || !(inpID in
datePickers)) return false; datePickers[inpID].disableDatePicker();},
                enable:                 function(inpID) { if(!inpID || !(inpID in
datePickers)) return false; datePickers[inpID].enableDatePicker();}
        };
})();
```

8.8.2 lib_actualizar_administradores.js

```
var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_login' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_password' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_nombres' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_ap_paterno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_ap_materno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

```
var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_tel_casa' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_direccion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_codigo_postal' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_email' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_genero' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_pais' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_administradores_administrador_perfil' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.3 lib_actualizar_casos.js

```
var v = new LiveValidation( 'administradores_actualizar_casos_caso_titulo' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_actualizar_casos_caso_contenido_caso'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.4 lib_actualizar_examenes.js

```
var v = new LiveValidation( 'administradores_actualizar_examenes_examen_titulo' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.5 lib_actualizar_medicos_adscritos.js

```
var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_login' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_password' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_nombres' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_ap_paterno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_ap_materno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_tel_casa' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_direccion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_codigo_postal' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_email' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_genero' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_pais' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_adscritos_medico_adscrito_perfil' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.6 lib_actualizar_medicos_residentes.js

```
var v = new LiveValidation(
```

```
'administradores_actualizar_medicos_residentes_medico_residente_login' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_password' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_nombres' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_ap_paterno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_ap_materno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_tel_casa' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_direccion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_codigo_postal' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_email' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_genero' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_pais' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_perfil' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_medicos_residentes_medico_residente_escuela' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.7 lib_actualizar_pacientes.js

```
var v = new LiveValidation( 'administradores_actualizar_pacientes_paciente_login'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_pacientes_paciente_password' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_pacientes_paciente_nombres' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_pacientes_paciente_ap_paterno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_pacientes_paciente_ap_materno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_pacientes_paciente_tel_casa' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_pacientes_paciente_direccion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_pacientes_paciente_codigo_postal' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_actualizar_pacientes_paciente_email'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_actualizar_pacientes_paciente_genero'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_actualizar_pacientes_paciente_pais'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_actualizar_pacientes_paciente_perfil'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_actualizar_pacientes_paciente_no_afiliacion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

```
var v = new LiveValidation(
'administradores_actualizar_pacientes_paciente_tipo_sangre' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.8 lib_insertar_administradores.js

```
var v = new LiveValidation(
'administradores_insertar_administradores_administrador_login' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_administradores_administrador_password' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_administradores_administrador_nombres' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_administradores_administrador_ap_paterno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_administradores_administrador_ap_materno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_administradores_administrador_tel_casa' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_administradores_administrador_direccion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_administradores_administrador_codigo_postal' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_administradores_administrador_email' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_administradores_administrador_genero' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
```

```
'administradores_insertar_administradores_administrador_pais' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.9 lib_insertar_casos.js

```
var v = new LiveValidation( 'administradores_insertar_casos_caso_titulo' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_insertar_casos_caso_contenido_caso'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );;
```

8.8.10 lib_insertar_contenidos.js

```
var v = new LiveValidation(
'administradores_insertar_contenidos_contenido_titulo_contenido' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_contenidos_contenido_descripcion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );;
```

8.8.11 lib_insertar_examenes.js

```
var v = new LiveValidation( 'administradores_insertar_examenes_examen_titulo' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.12 lib_insertar_medicos_adscritos.js

```
var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_login' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_password' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_nombres' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

```
var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_ap_paterno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_ap_materno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_tel_casa' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_direccion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_codigo_postal' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_email' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_genero' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_adscritos_medico_adscrito_pais' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.13 lib_insertar_medicos_residentes.js

```
var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_login' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_password' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_nombres' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_ap_paterno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
```

```
'administradores_insertar_medicos_residentes_medico_residente_ap_materno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_tel_casa' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_direccion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_codigo_postal' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_email' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_genero' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_pais' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_medicos_residentes_medico_residente_escuela' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.14 lib_insertar_pacientes.js

```
var v = new LiveValidation( 'administradores_insertar_pacientes_paciente_login' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_insertar_pacientes_paciente_password'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_insertar_pacientes_paciente_nombres'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_pacientes_paciente_ap_paterno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_pacientes_paciente_ap_materno' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

```
var v = new LiveValidation( 'administradores_insertar_pacientes_paciente_tel_casa'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_pacientes_paciente_direccion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_pacientes_paciente_codigo_postal' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_insertar_pacientes_paciente_email' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_insertar_pacientes_paciente_genero'
);
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation( 'administradores_insertar_pacientes_paciente_pais' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_pacientes_paciente_no_afiliacion' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );

var v = new LiveValidation(
'administradores_insertar_pacientes_paciente_tipo_sangre' );
v.add( Validate.Presence, {failureMessage: " Obligatorio"} );
```

8.8.15 livevalidation.js

```
// LiveValidation 1.3 (standalone version)
// Copyright (c) 2007-2008 Alec Hill (www.livevalidation.com)
// LiveValidation is licensed under the terms of the MIT License

/********************************************* LiveValidation class
*********************************/

/**
 *    validates a form field in real-time based on validations you assign to it
 *
 *    @var element {mixed} - either a dom element reference or the string id of the
element to validate
 *    @var optionsObj {Object} - general options, see below for details
 *
 *    optionsObj properties:
 *                                        validMessage {String}   - the message to
show when the field passes validation
 *
```

```
(DEFAULT: "Thankyou!")
 *                                      onValid {Function}           - function
to execute when field passes validation
 *
(DEFAULT: function(){ this.insertMessage(this.createMessageSpan());
this.addFieldClass(); } )
 *                                      onInvalid {Function}    - function to
execute when field fails validation
 *
(DEFAULT: function(){ this.insertMessage(this.createMessageSpan());
this.addFieldClass(); })
 *                                      insertAfterWhatNode {Int}     - position
to insert default message
 *
(DEFAULT: the field that is being validated)
 *              onlyOnBlur {Boolean} - whether you want it to validate as you type
or only on blur
 *                              (DEFAULT: false)
 *              wait {Integer} - the time you want it to pause from the last
keystroke before it validates (ms)
 *                              (DEFAULT: 0)
 *              onlyOnSubmit {Boolean} - whether should be validated only when the
form it belongs to is submitted
 *                              (DEFAULT: false)
 */
var LiveValidation = function(element, optionsObj){
      this.initialize(element, optionsObj);
}

LiveValidation.VERSION = '1.3 standalone';

/** element types constants ****/

LiveValidation.TEXTAREA           = 1;
LiveValidation.TEXT                   = 2;
LiveValidation.PASSWORD           = 3;
LiveValidation.CHECKBOX           = 4;
LiveValidation.SELECT = 5;
LiveValidation.FILE = 6;

/****** Static methods *******/

/**
 *    pass an array of LiveValidation objects and it will validate all of them
 *
 *    @var validations {Array} - an array of LiveValidation objects
 *    @return {Bool} - true if all passed validation, false if any fail

 */
LiveValidation.massValidate = function(validations){
  var returnValue = true;
      for(var i = 0, len = validations.length; i < len; ++i ){
            var valid = validations[i].validate();
            if(returnValue) returnValue = valid;
```

```
      }
      return returnValue;
}

/****** prototype ******/

LiveValidation.prototype = {

    validClass: 'LV_valid',
    invalidClass: 'LV_invalid',
    messageClass: 'LV_validation_message',
    validFieldClass: 'LV_valid_field',
    invalidFieldClass: 'LV_invalid_field',

    /**
     *      initialises all of the properties and events
     *
     * @var - Same as constructor above
     */
    initialize: function(element, optionsObj){
      var self = this;
      if(!element) throw new Error("LiveValidation::initialize - No element
reference or element id has been provided!");
      this.element = element.nodeName ? element : document.getElementById(element);
      if(!this.element) throw new Error("LiveValidation::initialize - No element
with reference or id of '" + element + "' exists!");
      // default properties that could not be initialised above
      this.validations = [];
      this.elementType = this.getElementType();
      this.form = this.element.form;
      // options
      var options = optionsObj || {};
      this.validMessage = options.validMessage || '';
      var node = options.insertAfterWhatNode || this.element;
            this.insertAfterWhatNode = node.nodeType ? node :
document.getElementById(node);
      this.onValid = options.onValid || function(){
this.insertMessage(this.createMessageSpan()); this.addFieldClass(); };
      this.onInvalid = options.onInvalid || function(){
this.insertMessage(this.createMessageSpan()); this.addFieldClass(); };
      this.onlyOnBlur =  options.onlyOnBlur || false;
      this.wait = options.wait || 0;
      this.onlyOnSubmit = options.onlyOnSubmit || false;
      // add to form if it has been provided
      if(this.form){
        this.formObj = LiveValidationForm.getInstance(this.form);
        this.formObj.addField(this);
      }
      // events
      // collect old events
      this.oldOnFocus = this.element.onfocus || function(){};
      this.oldOnBlur = this.element.onblur || function(){};
      this.oldOnClick = this.element.onclick || function(){};
      this.oldOnChange = this.element.onchange || function(){};
```

```
      this.oldOnKeyup = this.element.onkeyup || function(){};
      this.element.onfocus = function(e){ self.doOnFocus(e); return
self.oldOnFocus.call(this, e); };
      if(!this.onlyOnSubmit){
        switch(this.elementType){
          case LiveValidation.CHECKBOX:
            this.element.onclick = function(e){ self.validate(); return
self.oldOnClick.call(this, e); };
          // let it run into the next to add a change event too
          case LiveValidation.SELECT:
          case LiveValidation.FILE:
            this.element.onchange = function(e){ self.validate(); return
self.oldOnChange.call(this, e); };
            break;
          default:
            if(!this.onlyOnBlur) this.element.onkeyup = function(e){
self.deferValidation(); return self.oldOnKeyup.call(this, e); };
                this.element.onblur = function(e){ self.doOnBlur(e); return
self.oldOnBlur.call(this, e); };
        }
      }
    },

      /**
     *      destroys the instance's events (restoring previous ones) and removes it
from any LiveValidationForms
     */
    destroy: function(){
        if(this.formObj){
            // remove the field from the LiveValidationForm
            this.formObj.removeField(this);
            // destroy the LiveValidationForm if no LiveValidation fields left in
it
            this.formObj.destroy();
        }
      // remove events - set them back to the previous events
        this.element.onfocus = this.oldOnFocus;
      if(!this.onlyOnSubmit){
        switch(this.elementType){
          case LiveValidation.CHECKBOX:
            this.element.onclick = this.oldOnClick;
          // let it run into the next to add a change event too
          case LiveValidation.SELECT:
          case LiveValidation.FILE:
            this.element.onchange = this.oldOnChange;
            break;
          default:
            if(!this.onlyOnBlur) this.element.onkeyup = this.oldOnKeyup;
                this.element.onblur = this.oldOnBlur;
        }
      }
      this.validations = [];
        this.removeMessageAndFieldClass();
    },
```

```
    /**
     *      adds a validation to perform to a LiveValidation object
     *
     *      @var validationFunction {Function} - validation function to be used (ie
Validate.Presence )
     *      @var validationParamsObj {Object} - parameters for doing the
validation, if wanted or necessary
     * @return {Object} - the LiveValidation object itself so that calls can be
chained
     */
    add: function(validationFunction, validationParamsObj){
      this.validations.push( {type: validationFunction, params:
validationParamsObj || {} } );
      return this;
    },

      /**
     *      removes a validation from a LiveValidation object - must have exactly
the same arguments as used to add it
     *
     *      @var validationFunction {Function} - validation function to be used (ie
Validate.Presence )
     *      @var validationParamsObj {Object} - parameters for doing the
validation, if wanted or necessary
     * @return {Object} - the LiveValidation object itself so that calls can be
chained
     */
    remove: function(validationFunction, validationParamsObj){
        var found = false;
        for( var i = 0, len = this.validations.length; i < len; i++ ){
                  if( this.validations[i].type == validationFunction ){
                        if (this.validations[i].params == validationParamsObj) {
                              found = true;
                              break;
                        }
                  }
        }
      if(found) this.validations.splice(i,1);
      return this;
    },


    /**
     * makes the validation wait the alotted time from the last keystroke
     */
    deferValidation: function(e){
      if(this.wait >= 300) this.removeMessageAndFieldClass();
      var self = this;
      if(this.timeout) clearTimeout(self.timeout);
      this.timeout = setTimeout( function(){ self.validate(); }, self.wait);
    },

    /**
```

```
     * sets the focused flag to false when field loses focus
     */
    doOnBlur: function(e){
      this.focused = false;
      this.validate(e);
    },

    /**
     * sets the focused flag to true when field gains focus
     */
    doOnFocus: function(e){
      this.focused = true;
      this.removeMessageAndFieldClass();
    },

    /**
     *      gets the type of element, to check whether it is compatible
     *
     *      @var validationFunction {Function} - validation function to be used (ie
Validate.Presence )
     *      @var validationParamsObj {Object} - parameters for doing the
validation, if wanted or necessary
     */
    getElementType: function(){
      switch(true){
        case (this.element.nodeName.toUpperCase() == 'TEXTAREA'):
        return LiveValidation.TEXTAREA;
      case (this.element.nodeName.toUpperCase() == 'INPUT' &&
this.element.type.toUpperCase() == 'TEXT'):
        return LiveValidation.TEXT;
      case (this.element.nodeName.toUpperCase() == 'INPUT' &&
this.element.type.toUpperCase() == 'PASSWORD'):
        return LiveValidation.PASSWORD;
      case (this.element.nodeName.toUpperCase() == 'INPUT' &&
this.element.type.toUpperCase() == 'CHECKBOX'):
        return LiveValidation.CHECKBOX;
      case (this.element.nodeName.toUpperCase() == 'INPUT' &&
this.element.type.toUpperCase() == 'FILE'):
        return LiveValidation.FILE;
      case (this.element.nodeName.toUpperCase() == 'SELECT'):
        return LiveValidation.SELECT;
        case (this.element.nodeName.toUpperCase() == 'INPUT'):
            throw new Error('LiveValidation::getElementType - Cannot use
LiveValidation on an ' + this.element.type + ' input!');
        default:
            throw new Error('LiveValidation::getElementType - Element must be an
input, select, or textarea!');
      }
    },

    /**
     *      loops through all the validations added to the LiveValidation object
and checks them one by one
     *
```

```
     *      @var validationFunction {Function} - validation function to be used (ie
Validate.Presence )
     *      @var validationParamsObj {Object} - parameters for doing the
validation, if wanted or necessary
     * @return {Boolean} - whether the all the validations passed or if one failed
     */
    doValidations: function(){
            this.validationFailed = false;
            for(var i = 0, len = this.validations.length; i < len; ++i){
            var validation = this.validations[i];
            switch(validation.type){
                  case Validate.Presence:
                case Validate.Confirmation:
                case Validate.Acceptance:
                        this.displayMessageWhenEmpty = true;
                        this.validationFailed = !
this.validateElement(validation.type, validation.params);
                        break;
                  default:
                        this.validationFailed = !
this.validateElement(validation.type, validation.params);
                        break;
            }
            if(this.validationFailed) return false;
      }
      this.message = this.validMessage;
      return true;
    },

    /**
     *      performs validation on the element and handles any error (validation or
otherwise) it throws up
     *
     *      @var validationFunction {Function} - validation function to be used (ie
Validate.Presence )
     *      @var validationParamsObj {Object} - parameters for doing the
validation, if wanted or necessary
     * @return {Boolean} - whether the validation has passed or failed
     */
    validateElement: function(validationFunction, validationParamsObj){
            var value = (this.elementType == LiveValidation.SELECT) ?
this.element.options[this.element.selectedIndex].value : this.element.value;
        if(validationFunction == Validate.Acceptance){
          if(this.elementType != LiveValidation.CHECKBOX) throw new
Error('LiveValidation::validateElement - Element to validate acceptance must be a
checkbox!');
            value = this.element.checked;
      }
        var isValid = true;
            try{
            validationFunction(value, validationParamsObj);
      } catch(error) {
            if(error instanceof Validate.Error){
                  if( value !== '' || (value === '' &&
```

```
this.displayMessageWhenEmpty) ){
                        this.validationFailed = true;
                        this.message = error.message;
                        isValid = false;
                  }
            }else{
                  throw error;
            }
      }finally{
          return isValid;
        }
    },

    /**
     *      makes it do the all the validations and fires off the onValid or
onInvalid callbacks
     *
     * @return {Boolean} - whether the all the validations passed or if one failed
     */
    validate: function(){
      if(!this.element.disabled){
            var isValid = this.doValidations();
            if(isValid){
                  this.onValid();
                  return true;
            }else {
                  this.onInvalid();
                  return false;
            }
        }else{
      return true;
    }
    },

 /**
   *  enables the field
   *
   *  @return {LiveValidation} - the LiveValidation object for chaining
   */
  enable: function(){
      this.element.disabled = false;
      return this;
  },

  /**
   *  disables the field and removes any message and styles associated with the
field
   *
   *  @return {LiveValidation} - the LiveValidation object for chaining
   */
  disable: function(){
      this.element.disabled = true;
      this.removeMessageAndFieldClass();
      return this;
```

```
  },

    /** Message insertion methods ***************************
     *
     * These are only used in the onValid and onInvalid callback functions and so
if you overide the default callbacks,
     * you must either impliment your own functions to do whatever you want, or
call some of these from them if you
     * want to keep some of the functionality
     */

    /**
     *      makes a span containg the passed or failed message
     *
     * @return {HTMLSpanObject} - a span element with the message in it
     */
    createMessageSpan: function(){
        var span = document.createElement('span');
      var textNode = document.createTextNode(this.message);
            span.appendChild(textNode);
        return span;
    },

    /**
     *      inserts the element containing the message in place of the element that
already exists (if it does)
     *
     * @var elementToIsert {HTMLElementObject} - an element node to insert
     */
    insertMessage: function(elementToInsert){
            this.removeMessage();
            if( (this.displayMessageWhenEmpty && (this.elementType ==
LiveValidation.CHECKBOX || this.element.value == ''))
        || this.element.value != '' ){
            var className = this.validationFailed ? this.invalidClass :
this.validClass;
            elementToInsert.className += ' ' + this.messageClass + ' ' + className;
            if(this.insertAfterWhatNode.nextSibling +1000 ){

      this.insertAfterWhatNode.parentNode.insertBefore(elementToInsert,
this.insertAfterWhatNode.nextSibling);
            }else{

this.insertAfterWhatNode.parentNode.appendChild(elementToInsert);
          }
      }
    },


    /**
     *      changes the class of the field based on whether it is valid or not
     */
    addFieldClass: function(){
        this.removeFieldClass();
```

```
        if(!this.validationFailed){
            if(this.displayMessageWhenEmpty || this.element.value != ''){
                if(this.element.className.indexOf(this.validFieldClass) == -1)
this.element.className += ' ' + this.validFieldClass;
            }
        }else{
            if(this.element.className.indexOf(this.invalidFieldClass) == -1)
this.element.className += ' ' + this.invalidFieldClass;
        }
    },

    /**
     *      removes the message element if it exists, so that the new message will
replace it
     */
    removeMessage: function(){
      var nextEl;
      var el = this.insertAfterWhatNode;
      while(el.nextSibling){
          if(el.nextSibling.nodeType === 1){
                  nextEl = el.nextSibling;
                  break;
            }
            el = el.nextSibling;
      }
            if(nextEl && nextEl.className.indexOf(this.messageClass) != -1)
this.insertAfterWhatNode.parentNode.removeChild(nextEl);
    },

    /**
     *      removes the class that has been applied to the field to indicte if
valid or not
     */
    removeFieldClass: function(){
      if(this.element.className.indexOf(this.invalidFieldClass) != -1)
this.element.className =
this.element.className.split(this.invalidFieldClass).join('');
      if(this.element.className.indexOf(this.validFieldClass) != -1)
this.element.className = this.element.className.split(this.validFieldClass).join('
');
    },

    /**
     *      removes the message and the field class
     */
    removeMessageAndFieldClass: function(){
      this.removeMessage();
      this.removeFieldClass();
    }

} // end of LiveValidation class

/************************************ LiveValidationForm class
************************************/
```

```
/**
 * This class is used internally by LiveValidation class to associate a
LiveValidation field with a form it is icontained in one
 *
 * It will therefore not really ever be needed to be used directly by the
developer, unless they want to associate a LiveValidation
 * field with a form that it is not a child of
 */

/**
   *  handles validation of LiveValidation fields belonging to this form on its
submittal
   *
   *  @var element {HTMLFormElement} - a dom element reference to the form to turn
into a LiveValidationForm
   */
var LiveValidationForm = function(element){
  this.initialize(element);
}

/**
 * namespace to hold instances
 */
LiveValidationForm.instances = {};

/**
   *  gets the instance of the LiveValidationForm if it has already been made or
creates it if it doesnt exist
   *
   *  @var element {HTMLFormElement} - a dom element reference to a form
   */
LiveValidationForm.getInstance = function(element){
  var rand = Math.random() * Math.random();
  if(!element.id) element.id = 'formId_' + rand.toString().replace(/\./, '') + new
Date().valueOf();
  if(!LiveValidationForm.instances[element.id])
LiveValidationForm.instances[element.id] = new LiveValidationForm(element);
  return LiveValidationForm.instances[element.id];
}

LiveValidationForm.prototype = {

  /**
   *  constructor for LiveValidationForm - handles validation of LiveValidation
fields belonging to this form on its submittal
   *
   *  @var element {HTMLFormElement} - a dom element reference to the form to turn
into a LiveValidationForm
   */
  initialize: function(element){
      this.name = element.id;
    this.element = element;
    this.fields = [];
    // preserve the old onsubmit event
```

```
      this.oldOnSubmit = this.element.onsubmit || function(){};
    var self = this;
    this.element.onsubmit = function(e){
      return (LiveValidation.massValidate(self.fields)) ?
self.oldOnSubmit.call(this, e || window.event) !== false : false;
    }
  },

  /**
   *  adds a LiveValidation field to the forms fields array
   *
   *  @var element {LiveValidation} - a LiveValidation object
   */
  addField: function(newField){
    this.fields.push(newField);
  },

  /**
   *  removes a LiveValidation field from the forms fields array
   *
   *  @var victim {LiveValidation} - a LiveValidation object
   */
  removeField: function(victim){
      var victimless = [];
      for( var i = 0, len = this.fields.length; i < len; i++){
            if(this.fields[i] !== victim) victimless.push(this.fields[i]);
      }
    this.fields = victimless;
  },

  /**
   *  destroy this instance and its events
   *
   * @var force {Boolean} - whether to force the detruction even if there are
fields still associated
   */
  destroy: function(force){
      // only destroy if has no fields and not being forced
      if (this.fields.length != 0 && !force) return false;
      // remove events - set back to previous events
      this.element.onsubmit = this.oldOnSubmit;
      // remove from the instances namespace
      LiveValidationForm.instances[this.name] = null;
      return true;
  }

};// end of LiveValidationForm prototype

/********************************** Validate class
*********************************/
/**
 * This class contains all the methods needed for doing the actual validation
itself
 *
```

```
 * All methods are static so that they can be used outside the context of a form
field
 * as they could be useful for validating stuff anywhere you want really
 *
 * All of them will return true if the validation is successful, but will raise a
ValidationError if
 * they fail, so that this can be caught and the message explaining the error can
be accessed ( as just
 * returning false would leave you a bit in the dark as to why it failed )
 *
 * Can use validation methods alone and wrap in a try..catch statement yourself if
you want to access the failure
 * message and handle the error, or use the Validate::now method if you just want
true or false
 */

var Validate = {

    /**
     *      validates that the field has been filled in
     *
     *      @var value {mixed} - value to be checked
     *      @var paramsObj {Object} - parameters for this particular validation,
see below for details
     *
     *      paramsObj properties:
     *                                          failureMessage {String} - the
message to show when the field fails validation
     *
(DEFAULT: "Can't be empty!")
     */
    Presence: function(value, paramsObj){
            var paramsObj = paramsObj || {};
      var message = paramsObj.failureMessage || "Campo Obligatorio";
      if(value === '' || value === null || value === undefined){
            Validate.fail(message);
      }
      return true;
    },

    /**
     *      validates that the value is numeric, does not fall within a given range
of numbers
     *
     *      @var value {mixed} - value to be checked
     *      @var paramsObj {Object} - parameters for this particular validation,
see below for details
     *
     *      paramsObj properties:
     *                                          notANumberMessage {String} - the
message to show when the validation fails when value is not a number
     *
        (DEFAULT: "Must be a number!")
     *                                          notAnIntegerMessage {String} - the
```

```
message to show when the validation fails when value is not an integer
     *
        (DEFAULT: "Must be a number!")
     *                                          wrongNumberMessage {String} - the
message to show when the validation fails when is param is used
     *
        (DEFAULT: "Must be {is}!")
     *                                          tooLowMessage {String}       - the
message to show when the validation fails when minimum param is used
     *
        (DEFAULT: "Must not be less than {minimum}!")
     *                                          tooHighMessage {String}      - the
message to show when the validation fails when maximum param is used
     *
        (DEFAULT: "Must not be more than {maximum}!")
     *                                          is {Int}
      - the length must be this long
     *                                          minimum {Int}
      - the minimum length allowed
     *                                          maximum {Int}
      - the maximum length allowed
     *                          onlyInteger {Boolean} - if true will only allow
integers to be valid
     *                                                                  (DEFAULT:
false)
     *
     *  NB. can be checked if it is within a range by specifying both a minimum
and a maximum
     *  NB. will evaluate numbers represented in scientific form (ie 2e10)
correctly as numbers
     */
    Numericality: function(value, paramsObj){
        var suppliedValue = value;
        var value = Number(value);
      var paramsObj = paramsObj || {};
        var minimum = ((paramsObj.minimum) || (paramsObj.minimum == 0)) ?
paramsObj.minimum : null;;
        var maximum = ((paramsObj.maximum) || (paramsObj.maximum == 0)) ?
paramsObj.maximum : null;
      var is = ((paramsObj.is) || (paramsObj.is == 0)) ? paramsObj.is : null;
        var notANumberMessage = paramsObj.notANumberMessage || "Campo num\ufffdrico";
        var notAnIntegerMessage = paramsObj.notAnIntegerMessage || "Ingresa un
entero";
      var wrongNumberMessage = paramsObj.wrongNumberMessage || "Must be " + is +
"!";
      var tooLowMessage = paramsObj.tooLowMessage || "N\ufffdmero m\ufffdnimo: " + minimum;
      var tooHighMessage = paramsObj.tooHighMessage || "N\ufffdemro m\ufffdximo: " +
maximum;
        if (!isFinite(value)) Validate.fail(notANumberMessage);
        if (paramsObj.onlyInteger && (/\.0+$|\.$/.test(String(suppliedValue))  ||
value != parseInt(value)) ) Validate.fail(notAnIntegerMessage);
      switch(true){
            case (is !== null):
                  if( value != Number(is) ) Validate.fail(wrongNumberMessage);
```

```
                break;
            case (minimum !== null && maximum !== null):
                Validate.Numericality(value, {tooLowMessage: tooLowMessage,
minimum: minimum});
                Validate.Numericality(value, {tooHighMessage: tooHighMessage,
maximum: maximum});
                break;
            case (minimum !== null):
                if( value < Number(minimum) ) Validate.fail(tooLowMessage);
                break;
            case (maximum !== null):
                if( value > Number(maximum) ) Validate.fail(tooHighMessage);
                break;
      }
      return true;
    },

    /**
     *      validates against a RegExp pattern
     *
     *      @var value {mixed} - value to be checked
     *      @var paramsObj {Object} - parameters for this particular validation,
see below for details
     *
     *      paramsObj properties:
     *                                          failureMessage {String} - the
message to show when the field fails validation
     *
(DEFAULT: "Not valid!")
     *                                          pattern {RegExp}        - the
regular expression pattern
     *
(DEFAULT: /./)
     *              negate {Boolean} - if set to true, will validate true if the
pattern is not matched
   *                            (DEFAULT: false)
     *
     *  NB. will return true for an empty string, to allow for non-required, empty
fields to validate.
     *             If you do not want this to be the case then you must either add a
LiveValidation.PRESENCE validation
     *             or build it into the regular expression pattern
     */
    Format: function(value, paramsObj){
      var value = String(value);
      var paramsObj = paramsObj || {};
      var message = paramsObj.failureMessage || "Not valid!";
      var pattern = paramsObj.pattern || /./;
      var negate = paramsObj.negate || false;
      if(!negate && !pattern.test(value)) Validate.fail(message); // normal
      if(negate && pattern.test(value)) Validate.fail(message); // negated
      return true;
    },
```

```
    /**
     *      validates that the field contains a valid email address
     *
     *      @var value {mixed} - value to be checked
     *      @var paramsObj {Object} - parameters for this particular validation,
see below for details
     *
     *      paramsObj properties:
     *                                          failureMessage {String} - the
message to show when the field fails validation
     *
(DEFAULT: "Must be a number!" or "Must be an integer!")
     */
    Email: function(value, paramsObj){
      var paramsObj = paramsObj || {};
      var message = paramsObj.failureMessage || "Correo electr□nico inv□lido";
      Validate.Format(value, { failureMessage: message, pattern: /^([^@\s]+)@((?:
[-a-z0-9]+\.)+[a-z]{2,})$/i } );
      return true;
    },

    /**
     *      validates the length of the value
     *
     *      @var value {mixed} - value to be checked
     *      @var paramsObj {Object} - parameters for this particular validation,
see below for details
     *
     *      paramsObj properties:
     *                                          wrongLengthMessage {String} - the
message to show when the fails when is param is used
     *
        (DEFAULT: "Must be {is} characters long!")
     *                                          tooShortMessage {String}     - the
message to show when the fails when minimum param is used
     *
        (DEFAULT: "Must not be less than {minimum} characters long!")
     *                                          tooLongMessage {String}      - the
message to show when the fails when maximum param is used
     *
        (DEFAULT: "Must not be more than {maximum} characters long!")
     *                                          is {Int}
      - the length must be this long
     *                                          minimum {Int}
      - the minimum length allowed
     *                                          maximum {Int}
      - the maximum length allowed
     *
     *  NB. can be checked if it is within a range by specifying both a minimum
and a maximum
     */
    Length: function(value, paramsObj){
      var value = String(value);
      var paramsObj = paramsObj || {};
```

```
        var minimum = ((paramsObj.minimum) || (paramsObj.minimum == 0)) ?
paramsObj.minimum : null;
      var maximum = ((paramsObj.maximum) || (paramsObj.maximum == 0)) ?
paramsObj.maximum : null;
      var is = ((paramsObj.is) || (paramsObj.is == 0)) ? paramsObj.is : null;
        var wrongLengthMessage = paramsObj.wrongLengthMessage || "Ingresa " + is +
" caracteres";
      var tooShortMessage = paramsObj.tooShortMessage || "M⁄nimo " + minimum + "
caracteres";
      var tooLongMessage = paramsObj.tooLongMessage || "M⁄ximo " + maximum + "
caracteres";
      switch(true){
            case (is !== null):
                  if( value.length != Number(is) )
Validate.fail(wrongLengthMessage);
                  break;
            case (minimum !== null && maximum !== null):
                  Validate.Length(value, {tooShortMessage: tooShortMessage,
minimum: minimum});
                  Validate.Length(value, {tooLongMessage: tooLongMessage, maximum:
maximum});
                  break;
            case (minimum !== null):
                  if( value.length < Number(minimum) )
Validate.fail(tooShortMessage);
                  break;
            case (maximum !== null):
                  if( value.length > Number(maximum) )
Validate.fail(tooLongMessage);
                  break;
            default:
                  throw new Error("Validate::Length - Length(s) to validate against
must be provided!");
      }
      return true;
    },

    /**
     *      validates that the value falls within a given set of values
     *
     *      @var value {mixed} - value to be checked
     *      @var paramsObj {Object} - parameters for this particular validation,
see below for details
     *
     *      paramsObj properties:
     *                                          failureMessage {String} - the
message to show when the field fails validation
     *
(DEFAULT: "Must be included in the list!")
     *                                          within {Array}              - an
array of values that the value should fall in
     *
(DEFAULT: [])
     *                                          allowNull {Bool}       - if true,
```

```
and a null value is passed in, validates as true
     *
(DEFAULT: false)
     *             partialMatch {Bool}    - if true, will not only validate against
the whole value to check but also if it is a substring of the value
     *
(DEFAULT: false)
     *             caseSensitive {Bool} - if false will compare strings case
insensitively
     *                           (DEFAULT: true)
     *             negate {Bool}         - if true, will validate that the value
is not within the given set of values
     *
(DEFAULT: false)
     */
    Inclusion: function(value, paramsObj){
      var paramsObj = paramsObj || {};
      var message = paramsObj.failureMessage || "Must be included in the list!";
      var caseSensitive = (paramsObj.caseSensitive === false) ? false : true;
      if(paramsObj.allowNull && value == null) return true;
      if(!paramsObj.allowNull && value == null) Validate.fail(message);
      var within = paramsObj.within || [];
      //if case insensitive, make all strings in the array lowercase, and the
value too
      if(!caseSensitive){
        var lowerWithin = [];
        for(var j = 0, length = within.length; j < length; ++j){
            var item = within[j];
          if(typeof item == 'string') item = item.toLowerCase();
          lowerWithin.push(item);
        }
        within = lowerWithin;
        if(typeof value == 'string') value = value.toLowerCase();
      }
      var found = false;
      for(var i = 0, length = within.length; i < length; ++i){
        if(within[i] == value) found = true;
        if(paramsObj.partialMatch){
          if(value.indexOf(within[i]) != -1) found = true;
        }
      }
      if( (!paramsObj.negate && !found) || (paramsObj.negate && found) )
Validate.fail(message);
      return true;
    },

    /**
     *      validates that the value does not fall within a given set of values
     *
     *      @var value {mixed} - value to be checked
     *      @var paramsObj {Object} - parameters for this particular validation,
see below for details
     *
     *      paramsObj properties:
```

```
     *                                          failureMessage {String} - the
message to show when the field fails validation
     *
(DEFAULT: "Must not be included in the list!")
     *                                          within {Array}               - an
array of values that the value should not fall in
     *
(DEFAULT: [])
     *                                          allowNull {Bool}       - if true,
and a null value is passed in, validates as true
     *
(DEFAULT: false)
     *             partialMatch {Bool}    - if true, will not only validate against
the whole value to check but also if it is a substring of the value
     *
(DEFAULT: false)
     *             caseSensitive {Bool} - if false will compare strings case
insensitively
     *                           (DEFAULT: true)
     */
    Exclusion: function(value, paramsObj){
      var paramsObj = paramsObj || {};
      paramsObj.failureMessage = paramsObj.failureMessage || "Must not be included
in the list!";
      paramsObj.negate = true;
      Validate.Inclusion(value, paramsObj);
      return true;
    },

    /**
     *      validates that the value matches that in another field
     *
     *      @var value {mixed} - value to be checked
     *      @var paramsObj {Object} - parameters for this particular validation,
see below for details
     *
     *      paramsObj properties:
     *                                          failureMessage {String} - the
message to show when the field fails validation
     *
(DEFAULT: "Does not match!")
     *                                          match {String}               - id
of the field that this one should match
     */
    Confirmation: function(value, paramsObj){
            if(!paramsObj.match) throw new Error("Validate::Confirmation - Error
validating confirmation: Id of element to match must be provided!");
      var paramsObj = paramsObj || {};
      var message = paramsObj.failureMessage || "Does not match!";
      var match = paramsObj.match.nodeName ? paramsObj.match :
document.getElementById(paramsObj.match);
      if(!match) throw new Error("Validate::Confirmation - There is no reference
with name of, or element with id of '" + paramsObj.match + "'!");
      if(value != match.value){
```

```
            Validate.fail(message);
      }
      return true;
    },

    Menor: function(value, paramsObj){
            if(!paramsObj.match) throw new Error("Validate::Confirmation - Error
validating confirmation: Id of element to match must be provided!");
      var paramsObj = paramsObj || {};
      var message = paramsObj.failureMessage || "El numero es mayor";
      var match = paramsObj.match.nodeName ? paramsObj.match :
document.getElementById(paramsObj.match);
      if(!match) throw new Error("Validate::Confirmation - There is no reference
with name of, or element with id of '" + paramsObj.match + "'!");
      if( parseInt(value) > parseInt(match.value)){
            Validate.fail(message);
      }
      return true;
    },

    /**
     *      validates that the value is true (for use primarily in detemining if a
checkbox has been checked)
     *
     *      @var value {mixed} - value to be checked if true or not (usually a
boolean from the checked value of a checkbox)
     *      @var paramsObj {Object} - parameters for this particular validation,
see below for details
     *
     *      paramsObj properties:
     *                                          failureMessage {String} - the
message to show when the field fails validation
     *
(DEFAULT: "Must be accepted!")
     */
    Acceptance: function(value, paramsObj){
            var paramsObj = paramsObj || {};
      var message = paramsObj.failureMessage || "Must be accepted!";
      if(!value){
            Validate.fail(message);
      }
      return true;
    },

      /**
     *      validates against a custom function that returns true or false (or
throws a Validate.Error) when passed the value
     *
     *      @var value {mixed} - value to be checked
     *      @var paramsObj {Object} - parameters for this particular validation,
see below for details
     *
     *      paramsObj properties:
     *                                          failureMessage {String} - the
```

```
message to show when the field fails validation
     *
(DEFAULT: "Not valid!")
     *                                                against {Function}
      - a function that will take the value and object of arguments and return true
or false
     *
(DEFAULT: function(){ return true; })
     *                                                args {Object}          - an object
of named arguments that will be passed to the custom function so are accessible
through this object within it
     *
(DEFAULT: {})
     */
      Custom: function(value, paramsObj){
            var paramsObj = paramsObj || {};
            var against = paramsObj.against || function(){ return true; };
            var args = paramsObj.args || {};
            var message = paramsObj.failureMessage || "Not valid!";
          if(!against(value, args)) Validate.fail(message);
          return true;
        },

    /**
     *      validates whatever it is you pass in, and handles the validation error
for you so it gives a nice true or false reply
     *
     *      @var validationFunction {Function} - validation function to be used (ie
Validation.validatePresence )
     *      @var value {mixed} - value to be checked if true or not (usually a
boolean from the checked value of a checkbox)
     *      @var validationParamsObj {Object} - parameters for doing the
validation, if wanted or necessary
     */
    now: function(validationFunction, value, validationParamsObj){
            if(!validationFunction) throw new Error("Validate::now - Validation
function must be provided!");
      var isValid = true;
        try{
            validationFunction(value, validationParamsObj || {});
      } catch(error) {
            if(error instanceof Validate.Error){
                  isValid =  false;
            }else{
                  throw error;
            }
      }finally{
            return isValid
      }
    },

    /**
     * shortcut for failing throwing a validation error
     *
```

```
     *      @var errorMessage {String} - message to display
     */
    fail: function(errorMessage){
            throw new Validate.Error(errorMessage);
    },

    Error: function(errorMessage){
      this.message = errorMessage;
      this.name = 'ValidationError';
    }

}
```

9 Conclusiones

Durante la elaboración del proyecto terminal, se tuvo que ir liberando poco a poco cada módulo, se intentó documentar primero y programar después, como un buen programador lo haría, sin embargo, la falta de experiencia en el diseño y codificación de sistemas grandes empezó a pesar, ya que se observaban nuevos requerimientos confirme se iba codificando. El proyecto se retrasó un poco debido a esta falta de experiencia (de experiencia no de conocimiento) por parte del alumno, pero resulta bastante útil si el alumno (yo) decidiera elegir este camino de programador/diseñador/analista en un futuro.

El asesor intervino adecuadamente cuando se tuvieron dudas, recomendando referencias, cursos, libros. También poniendo su conocimiento en este Proyecto Terminal, estando disponible siempre que se le solicitó ayuda.

10 Referencias

http://static.springsource.org/spring/docs/3.1.x/spring-framework-reference/html/

Última fecha de consulta: 25/abril/2012

http://docs.jboss.org/hibernate/orm/4.1/manual/en-US/html/

Última fecha de consulta: 25/Abril/2012

http://struts.apache.org/2.x/docs/tutorials.html

Última fecha de consulta: 25/Abril/2012

http://struts.apache.org/2.x/docs/guides.html

Última fecha de consulta: 25/Abril/2012

http://dev.mysql.com/doc/refman/5.0/es/index.html

Última fecha de consulta: 25/Abril/2012

Julio 2009 – **Diplomado: Taller de Programación de Aplicaciones Web sobre plataforma JAVA.**

Impartido por: Ing. Mónica Silva, de Acsinet S.A. de C.V.

Septiembre 2009 – **Diplomado: Taller de Creación de Aplicaciones Web con Struts Action FrameWork**.

Impartido por: Ing. Mónica Silva, de Acsinet S.A. de C.V.

Octubre 2009 – **Diplomado: Taller de Lógica de Negocios con Java (Spring Framework / Hibernate)**

Impartido por: Ing. Mónica Silva de Acsinet S.A. de C.V.

Noviembre 2009 – **Diplomado: Taller de Servicios**

Impartido por: Ing. Mónica Silva de Acsinet S.A. de C.V.