Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Ingeniería en Computación

Proyecto Terminal II

"SISTEMA DE APOYO EMOCIONAL PARA PACIENTES DE ENFERMEDADES NEOPLÁSICAS"

Elaborado por:

González Hernández Rodrigo Daniel (204359136)

Trimestre 14-I

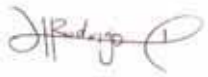
09 de Abril del 2014

Asesorado por: M. en C. Rafaela Blanca Silva López

Yo, M.C. Rafaela Blanca Silvia López, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

M.C. Rafaela Blanca Silvia López

Yo, Rodolfo Salomón Sánchez, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Objetivo general

Desarrollar el módulo del Sistema de E-Salud denominado: "Sistema de apoyo emocional para pacientes de enfermedades Neoplásicas".

Objetivos particulares

- Determinar las Necesidades de Pacientes con Enfermedades neoplásicas que requieren apoyo emocional.
- Diseñar el "Sistema de apoyo emocional para pacientes de enfermedades Neoplásicas que cubra los siguientes módulos:
  - Consulta de temas relacionados con enfermedades Neoplásicas.
  - Resolución de dudas de pacientes con especialistas u otros pacientes.
  - Foro de opiniones para pacientes.
- Implementar el Sistema de apoyo emocional para pacientes de enfermedades Neoplásicas.
- Realizar pruebas para la liberación del Sistema de apoyo emocional para pacientes de enfermedades Neoplásicas.
- Elaborar la documentación del Sistema de apoyo emocional para pacientes de enfermedades Neoplásicas.
- Integrar el Sistema de apoyo emocional para pacientes de enfermedades Neoplásicas, en los Ambientes Virtuales de Aprendizaje E-Learning – Knowledge[1]. [I]

Antecedentes:

Los soportes que brindan las Nuevas Tecnologías son poderosas herramientas para la diversificación de aplicaciones en cursos en línea, capacitación en sedes remotas, asesoría especializada, actividades académicas diversas de investigación, docencia y de autoaprendizaje, pero su incorporación requiere de planeación, seguimiento y evaluación.

En estas redes o cadenas electrónicas constituidas por personas con intereses comunes, se abren posibilidades de enseñar, de aprender, de actuar profesionalmente donde los involucrados se identifican en función de las finalidades con que participan, existiendo entonces, las de tipo pedagógico, de información pública y de información institucional, mediante la presentación de planes y servicios educativos o de difusión del conocimiento.

[1] El E-Learning-Knowledge es una plataforma de enseñanza – aprendizaje, que permite la constante evaluación y el fomento de conocimientos y habilidades en los alumnos.

En la actualidad, existe un programa de apoyo emocional en línea llamado “Your shoes” del “Breast Cancer Network of Strength”anteriormente conocido como “Y-ME National Breast Cancer Organization ®”; que se dedica a recabar fondos para financiar *Your Shoes*, programas de divulgación, talleres de sensibilización acerca de la salud de mama, bancos de pelucas y prótesis para las mujeres con recursos limitados y la promoción de las políticas relacionadas con el cáncer de mama. [II]

Your shoes, es un sistema similar al que se desea construir, que se encarga de apoyar a mujeres sobrevivientes del cáncer de mama. Este cuenta con una línea telefónica de asistencia las 24 horas del día, los 7 días de la semana. Una desventaja, es el requerir de personal calificado, que se encuentre disponible para atender a los usuarios en cualquier momento. Figura 1.0

Figura 1.0 “*Your Shoes”*

El sistema proporciona información referente al cáncer de mama, como se muestra en la Figura 1.1.

Figura 1.1 Información

Este programa cuenta con un foro, donde se tocan ciertos temas referentes al cáncer de mama, pero las personas que dejan sus comentarios lo hacen en inglés; esa es una desventaja que tiene este sistema, ya que no existe un foro especialmente para personas que sólo hablen español, la Figura 1.2 muestra esta desventaja.

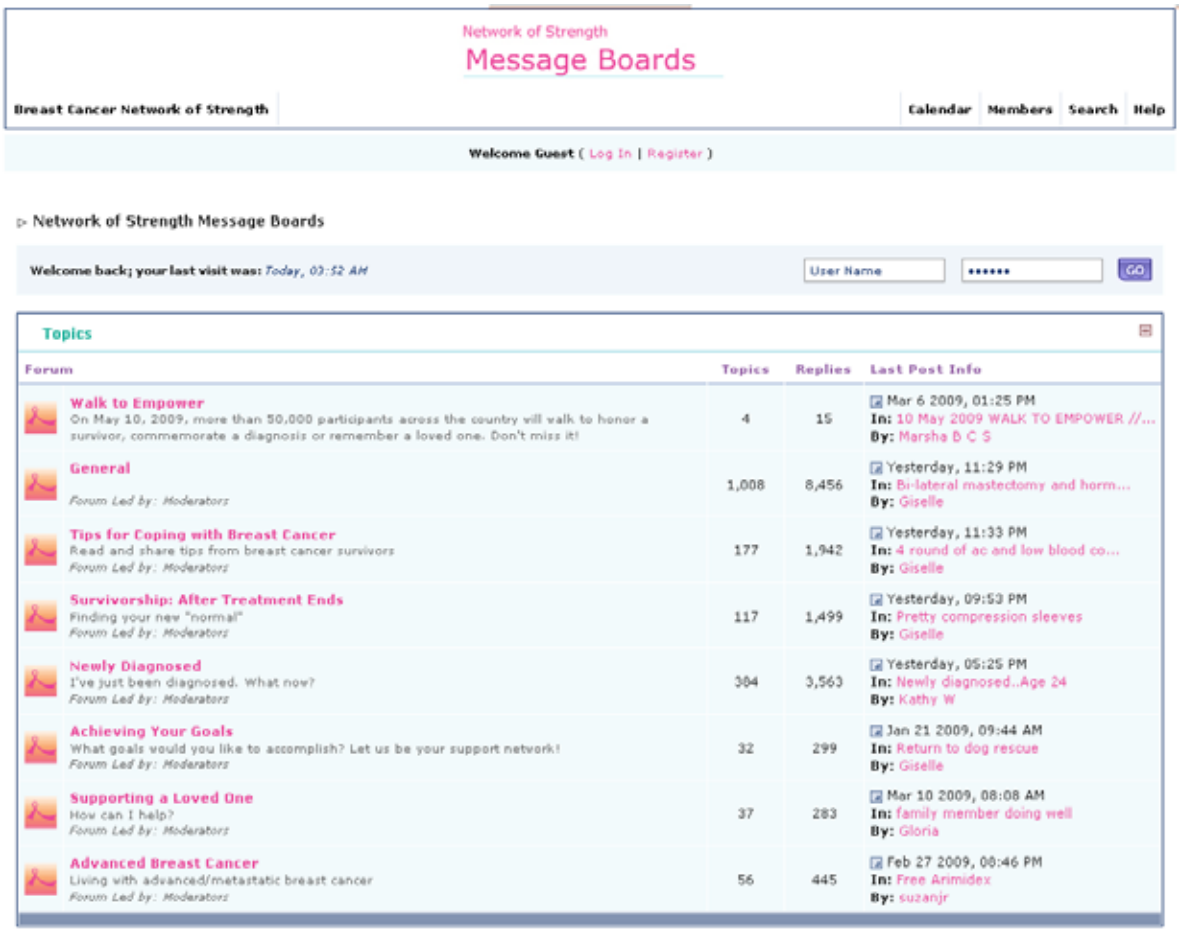


Figura 1.2 Foro

Ya que este sistema está enfocado a mujeres con diagnostico de cáncer de mama, se excluye a los demás pacientes con enfermedades neoplásicas.

Justificación:

Algunos sistemas de apoyo emocional existentes hoy en día, están enfocados a algún tipo de enfermedad específica. Es por eso que se desea realizar un sistema que cuente con un espacio para brindar apoyo emocional a cualquier tipo de paciente que sea diagnosticado con algún tipo de tumor.

Una neoplasia (llamada también tumor o blastoma) es una masa anormal de tejido, producida por la multiplicación de algún tipo de células; esta multiplicación es descoordinada con los mecanismos que controlan la multiplicación celular en el organismo, y los supera. Además, estos tumores, una vez originados, continúan creciendo aunque dejen de actuar las causas que los provocan. La neoplasia se conoce en general con el nombre de cáncer. [III]

Muchos de los sistemas de apoyo emocional existentes, solo cuentan con ayuda vía telefónica para asesorar a personas enfermas, esto hace difícil recibir asesoría a cualquier hora del día o que los pacientes la obtengan de personas realmente calificadas. Otro problema que se presenta, es que algunos de los sistemas que ya existen tienen cede en otros países o cuentan con otro idioma y esto dificulta el acceso a ellos.

Es importante que un paciente que enfrenta este tipo de enfermedades pueda consultar información relevante de manera rápida y sencilla, que le proporcione ayuda para encarar la situación que atraviesa, por lo cual, el sistema que se realizará, contará con un área especial de información que provea artículos relevantes para los usuarios.

Debido a esto, se pretende tener un espacio dedicado especialmente, para que los usuarios realicen preguntas o consultas a doctores especializados en el tema.

Otro punto importante que se desea cubrir, es que los pacientes puedan compartir opiniones entre ellos, ya que puede darse la posibilidad de que encuentren personas que ya hayan pasado por las mismas circunstancias y logren auxiliar a otros pacientes. Para ello se creará un foro donde los pacientes puedan dejar preguntas, mismas que puedan ser resueltas por otros pacientes o personal capacitado.

El sistema llamado “Your Shoes” posee, a manera de ayuda, un foro en línea, solo que este únicamente esta enfocado a mujeres con cáncer de mama. Excluyendo entonces, a todos los demás pacientes con enfermedades neoplásicas.

Otro problema con el que nos enfrentamos, es que algunos sistemas que si cuentan con un foro abierto para presentar dudas y opiniones, se realizan en otros idiomas, para esto, se llevará a cabo uno exclusivamente en español.

A lo largo de la carrera de Ingeniería en Computación dentro de la Universidad Autónoma Metropolitana Unidad Azcapotzalco, el alumno ha adquirido y aplicado

los conocimientos en diversas UEA´s para su aplicación directa a problemas. En la realización del Proyecto Terminal, se requerirá de los conocimientos avanzados de las UEA´s fundamentales para la correcta proyección, como: Bases de Datos, Estructura de Datos con Programación orientada a Objetos, Metodologías de Análisis Y Diseño de Sistemas de Información e Ingeniería de Software dedicados a la planeación, documentación y realización correcta de la proyección planteada.

Descripción técnica

El proyecto considera los siguientes escenarios:

- Módulo de gestión de contenidos
- Módulo de gestión de consultas
- Módulo del foro

Módulo de gestión de contenidos

El módulo de gestión de contenidos esta diseñado para que los usuarios o pacientes registrados al sistema, puedan tener acceso a archivos de información relevante que sirvan para resolver dudas sobre: Tratamientos, estudios clínicos, técnicas de diagnostico, procedimientos, entre otras cosas. Esto, con la finalidad de que los propios pacientes encuentren material de apoyo al que puedan acceder sin ningún problema.

El administrador será capaz de agregar, modificar, borrar y consultar el material. También existe la posibilidad de que un Médico efectúe las mismas operaciones que el Administrador, solo que este no podrá borrar el contenido. El usuario o paciente solo tendrá la opción de consultar el material. Figura 2.0

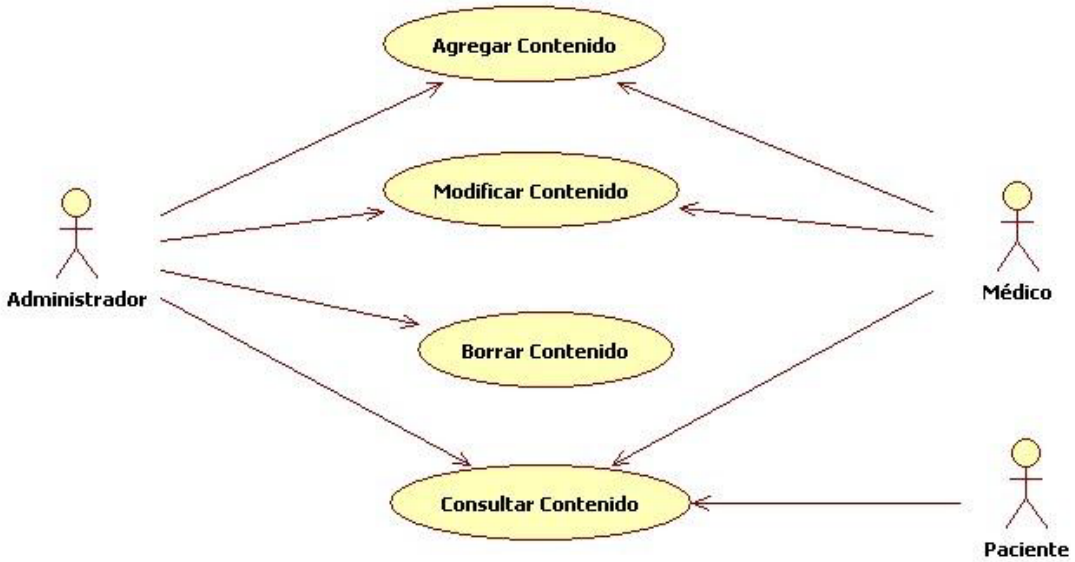


Figura 2.0 Diagrama de Casos de uso del Módulo de gestión de contenidos.

Módulo de gestión de consultas

Este módulo esta hecho para que los pacientes o usuarios puedan hacer preguntas a un médico. Los pacientes o usuarios, tendrán tanto la posibilidad de hacer nuevas preguntas, consultar las preguntas que otros usuarios hayan hecho y ver las respuestas que los médicos les dieron a dichas preguntas.

El administrador será capaz de crear el espacio para que se puedan agregar, consultar, contestar y borrar preguntas así como consultar y borrar respuestas posteriormente.

El médico tendrá las funciones de consultar las preguntas en cualquier momento y de responder las preguntas realizadas por los usuarios o pacientes. Figura 2.1

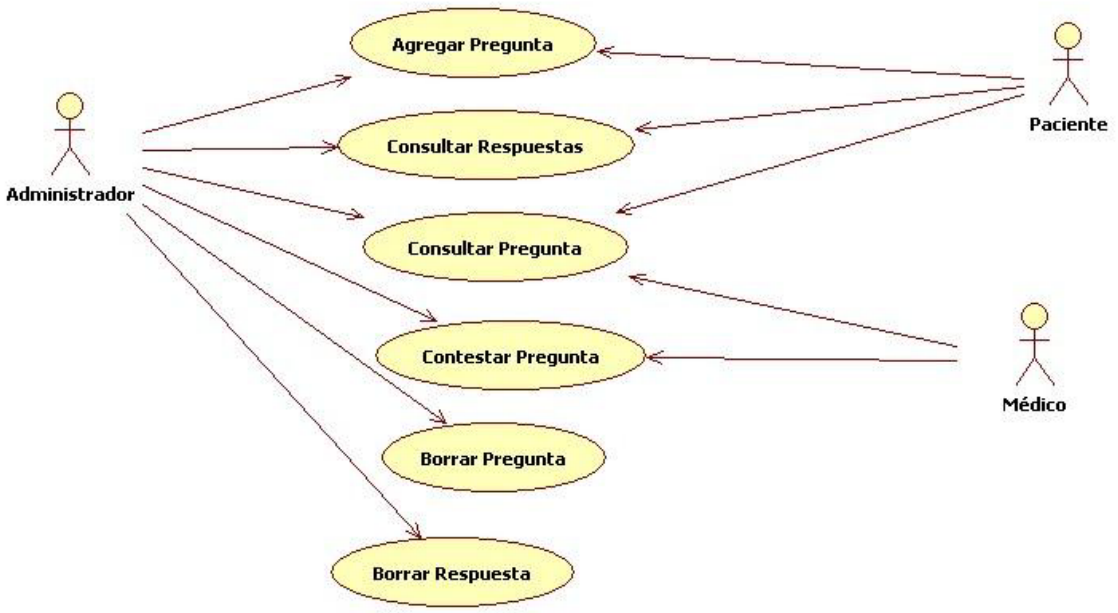


Figura 2.1 Diagrama de Casos de uso del Módulo de gestión de consultas.

Módulo de gestión de foro

El objeto con el que fue creado este módulo, es tener el control de un foro, en el cual, los pacientes o usuarios podrán externar sus opiniones o dudas dejando sus comentarios en este espacio.

El administrador poseerá la capacidad de agregar tema, consultar tema, borrar tema, agregar comentario y borrar comentario.

Los usuarios contarán con los privilegios para agregar un tema nuevo al foro, consultar los temas ya existentes y agregar uno o más comentarios a cada uno de los temas disponibles.

El médico solo contará con la posibilidad de realizar observaciones sobre los comentarios de otros usuarios o pacientes en el foro. Figura 2.2

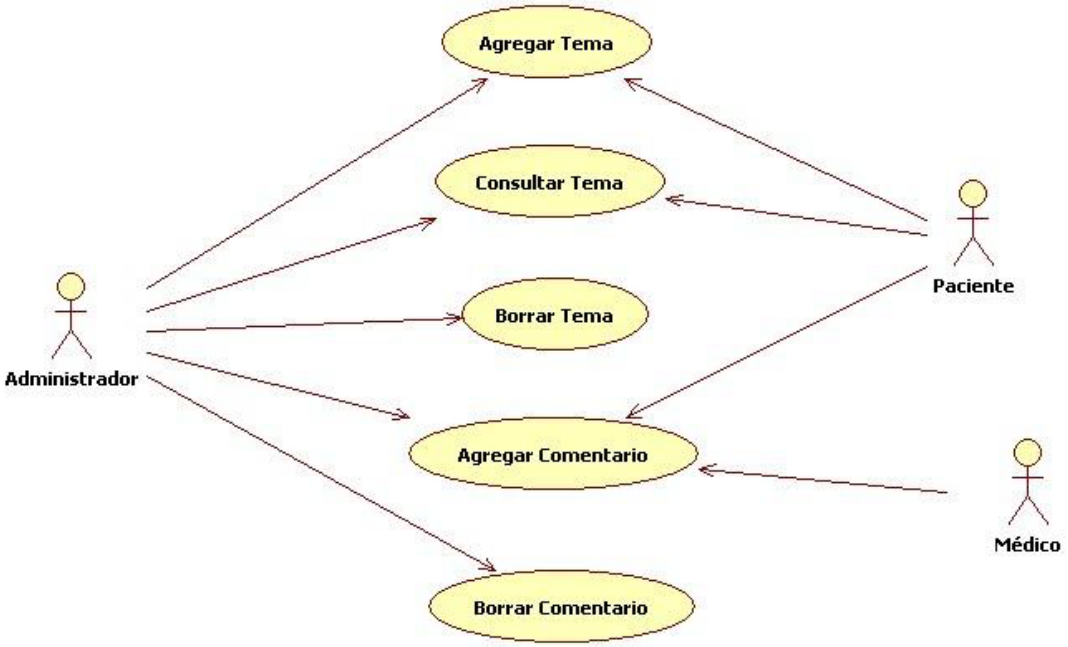


Figura 2.2 Diagrama de Casos de uso del Módulo de gestión de Foro.

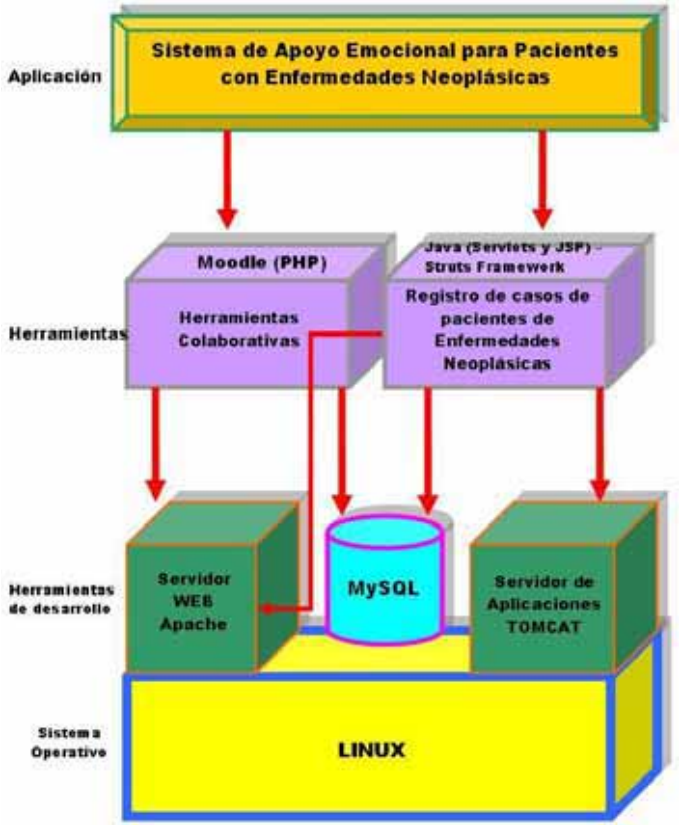
Especificaciones Técnicas.

El proyecto se desarrollará considerando la siguiente infraestructura tecnológica:

- Plataforma Linux
- Tecnologías Java para el desarrollo de aplicaciones WEB[2]. [IIIV]
    - ❖ Framework[3] Struts[4]. [V]
    - ❖ JSP[5]. [VI]
    - ❖ Servlets[6].
- Herramientas para sistemas de aprendizaje en línea
    - ❖ Moodle[7]
- Manejador de Base de Datos MySQL[8]
- Servidor WEB Apache. [VII]
- Servidor de aplicaciones TOMCAT. [VIII]

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

[2] Aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador.
[3] Es la extensión de un lenguaje mediante una o mas jerarquías de clases que implementan una funcionalidad y que opcionalmente pueden ser extendidas.
[4] Es un *framework* de la capa de presentación que implementa el patrón de patrón MVC (Modelo-Vista-Controlador) en Java.
[5] JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.



Las características mínimas que deberá tener el proyecto para darlo concluido serán:

- Un Sistema capaz de registrar los casos de los pacientes de enfermedades neoplásicas desde cualquier computadora con acceso a Internet y un navegador *WEB* actual.
- El Sistema Colaborativo permite consultar temas relacionados con enfermedades Neoplásicas y la resolución de dudas con especialistas u otros pacientes.

Al finalizar el proyecto se entregarán los siguientes documentos en un CD:
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.

[6] Pequeño programa que corre en un servidor. Por lo general son aplicaciones Java que corren en un entorno de servidor Web.
[7] Sistema de gestión de cursos, de distribución libre, que ayuda a los educadores a crear comunidades de aprendizaje en línea.
[8] Es un sistema de gestión de base de datos relacional, multihilo y multiusuario

Recursos Tecnológicos

El equipo utilizado requerirá:

- 256 MB de RAM
- 40 GB de HD
- Procesador Intel Pentium IV o superior
- Conexión a Internet
- Sistema Operativo Linux distribución Ubuntu 6.10 o superior.
- Software: MySQL, Apache, TOMCAT, Moodle.

Este equipo no es especializado y se tiene disponible al momento.

Equipo que se utilizará como servidor:

- 4 GB de RAM
- 100 GB de HD
- Intel Core Duo (2 procesadores)
- S.O. Linux
- Debe tener instalado: Servidor WEB Apache, TOMCAT, manejador de bases de datos MySQL y Herramientas de E-Learning.

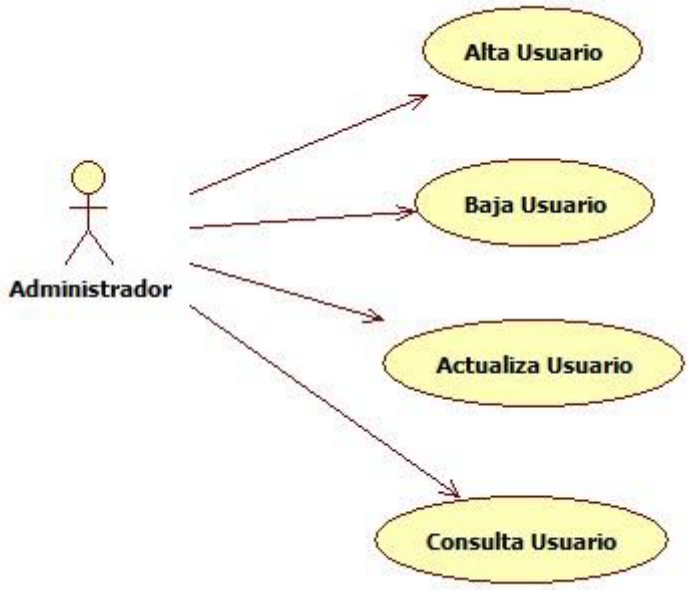
Este equipo especializado será proporcionado por la profesora Blanca Silva.

Desarrollo del Sistema de Apoyo para Pacientes con Enfermedades Neoplásicas

Diseño y Modelado del Sistema

Casos de Uso

Usuarios



Caso de Uso:	Alta Usuario
Descripción:	Modelar los escenarios que contemplan el registro de usuarios al sistema
Actores:	Usuario Administrador que puede dar de alta a un usuario.
Precondiciones:	Que el Administrador exista en la base y se pueda firmar en el sistema exitosamente.
Escenario Principal	El Administrador ingresa a la forma para dar de alta a un usuario y llena los campos correspondientes del usuario.

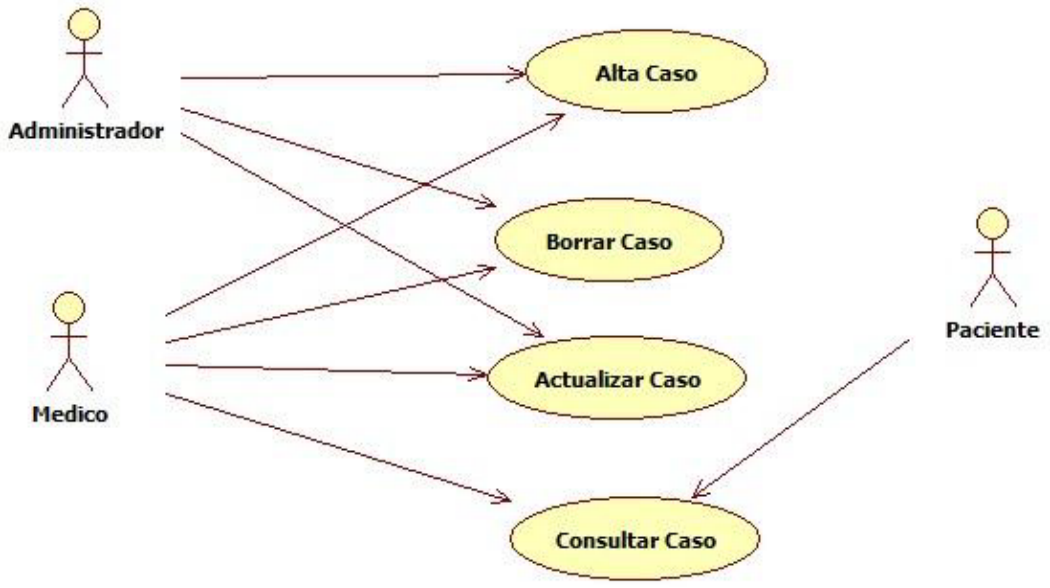
Escenario alternativo:	Ninguno
Poscondiciones:	Se da de alta un usuario exitosamente

Caso de Uso:	Baja Usuario
Descripción:	Modelar los escenarios que contemplan el borrado de usuarios al sistema
Actores:	Administrador. Administrador que desee borrar a un usuario.
Precondiciones:	Debe existir el usuario en el sistema.
Escenario Principal	El Administrador ingresa a la forma para dar de baja un usuario y borra al usuario correspondiente.
Escenario alternativo:	Ninguno
Poscondiciones:	Se borra un usuario exitosamente.

Caso de Uso:	Actualiza Usuario
Descripción:	Modelar los escenarios que contemplan la modificación de los datos de un usuario en el sistema.
Actores:	Administrador. Administrador que desee modificar la información de un usuario.
Precondiciones:	Debe existir el usuario en el sistema.
Escenario Principal	El Administrador ingresa a la forma para modificar a un usuario y actualiza los datos del usuario correspondiente.
Escenario alternativo:	Ninguno
Poscondiciones:	Se modifica un usuario exitosamente.

Caso de Uso:	Consultar Usuario
Descripción:	Modelar los escenarios que contemplan la consulta de los datos de un usuario en el sistema.
Actores:	Administrador. Administrador que desee consultar la información de un usuario.
Precondiciones:	Debe existir el usuario en el sistema.
Escenario Principal	El Administrador ingresa a la forma para consultar la información de un usuario en el sistema.
Escenario alternativo:	Ninguno
Poscondiciones:	Se realiza una consulta de un usuario exitosamente.

Casos Médicos



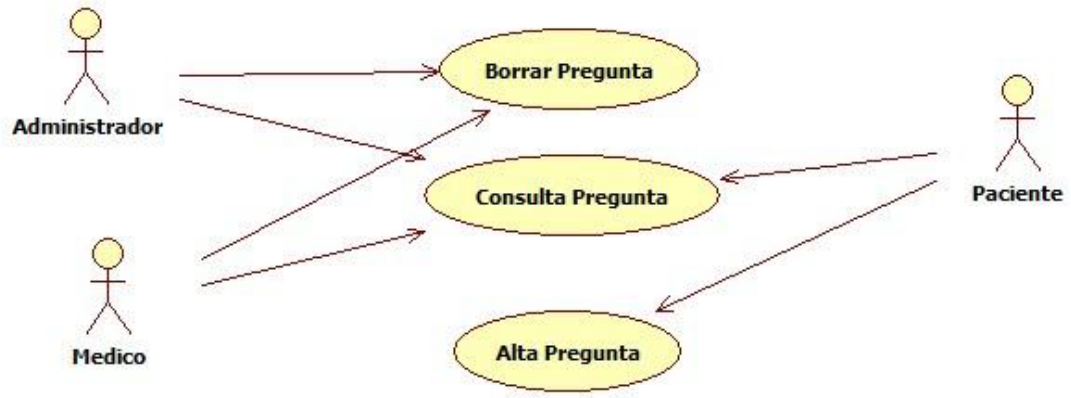
Caso de Uso:	Alta Caso
Descripción:	Modelar los escenarios que contemplan el registro de un caso médico nuevo al sistema.
Actores:	Administrador, Médico: Usuario que puede dar de alta a un nuevo caso.
Precondiciones:	Que el Administrador o el Médico estén firmados en el sistema
Escenario Principal	El Administrador o el Médico ingresan a la forma para dar de alta un caso médico nuevo y llena la descripción correspondiente.
Escenario alternativo:	Ninguno
Poscondiciones:	Se agrega un caso nuevo exitosamente.

Caso de Uso:	Baja Caso
Descripción:	Modelar los escenarios que contemplan el borrado de un caso médico del sistema.
Actores:	Administrador, Médico. Administrador o Médico que desee borrar un caso médico en el sistema.
Precondiciones:	Debe existir el caso médico en el sistema.
Escenario Principal	El Administrador o Médico ingresa a la forma para dar de baja un caso médico y lo borra de la lista correspondiente.
Escenario alternativo:	Ninguno
Poscondiciones:	Se borra un caso médico exitosamente.

Caso de Uso:	Actualiza Caso
Descripción:	Modelar los escenarios que contemplan la modificación de un caso médico en el sistema.
Actores:	Administrador, Médico. Administrador o Médico que desee modificar la información de un caso médico.
Precondiciones:	Debe existir el caso médico en el sistema.
Escenario Principal	El Administrador o Médico ingresa a la forma para modificar un caso médico y actualiza los campos correspondientes.
Escenario alternativo:	Ninguno
Poscondiciones:	Se modifica un caso médico exitosamente.

Caso de Uso:	Consultar Caso
Descripción:	Modelar los escenarios que contemplan la consulta de los datos de un caso médico en el sistema.
Actores:	Médico, Paciente. Médico o Paciente que desee consultar la información de un caso médico.
Precondiciones:	Debe existir el caso médico en el sistema.
Escenario Principal	El Médico o Paciente ingresa a la forma para consultar un caso médico.
Escenario alternativo:	Ninguno
Poscondiciones:	Se consulta el caso médico exitosamente.

Preguntas

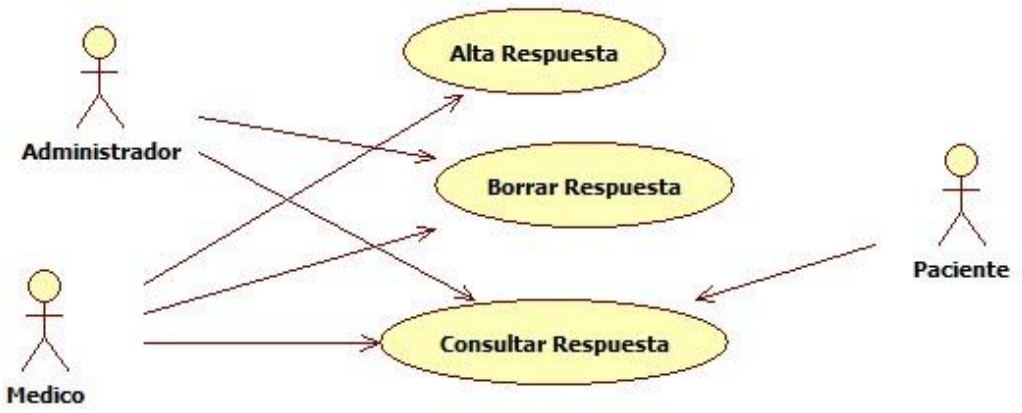


Caso de Uso:	Alta Pregunta
Descripción:	Modelar los escenarios que contemplan el registro de una pregunta en el sistema.
Actores:	Paciente: Usuario que puede dar de alta una pregunta.
Precondiciones:	Que el Paciente estén firmado en el sistema
Escenario Principal	El Paciente ingresa a la forma para dar de alta una pregunta y llena el campo correspondiente.
Escenario alternativo:	Ninguno
Poscondiciones:	Se agrega una pregunta exitosamente.

Caso de Uso:	Borrar Pregunta
Descripción:	Modelar los escenarios que contemplan el borrado de una pregunta del sistema.
Actores:	Administrador, Médico. Administrador o Médico que desee borrar una pregunta en el sistema.
Precondiciones:	Debe existir la pregunta en el sistema.
Escenario Principal	El Administrador o Médico ingresa a la forma para dar de baja una pregunta y la borra de la lista correspondiente.
Escenario alternativo:	Ninguno
Poscondiciones:	Se borra una pregunta exitosamente.

Caso de Uso:	Consultar Pregunta
Descripción:	Modelar los escenarios que contemplan la consulta de los datos de una pregunta en el sistema.
Actores:	Administrador, Médico, Paciente. Administrador, Médico o Paciente que desee consultar una pregunta.
Precondiciones:	Debe existir la pregunta en el sistema.
Escenario Principal	El Administrador, Médico o Paciente ingresa a la forma para consultar una pregunta.
Escenario alternativo:	Ninguno
Poscondiciones:	Se consulta una pregunta exitosamente.

Respuestas

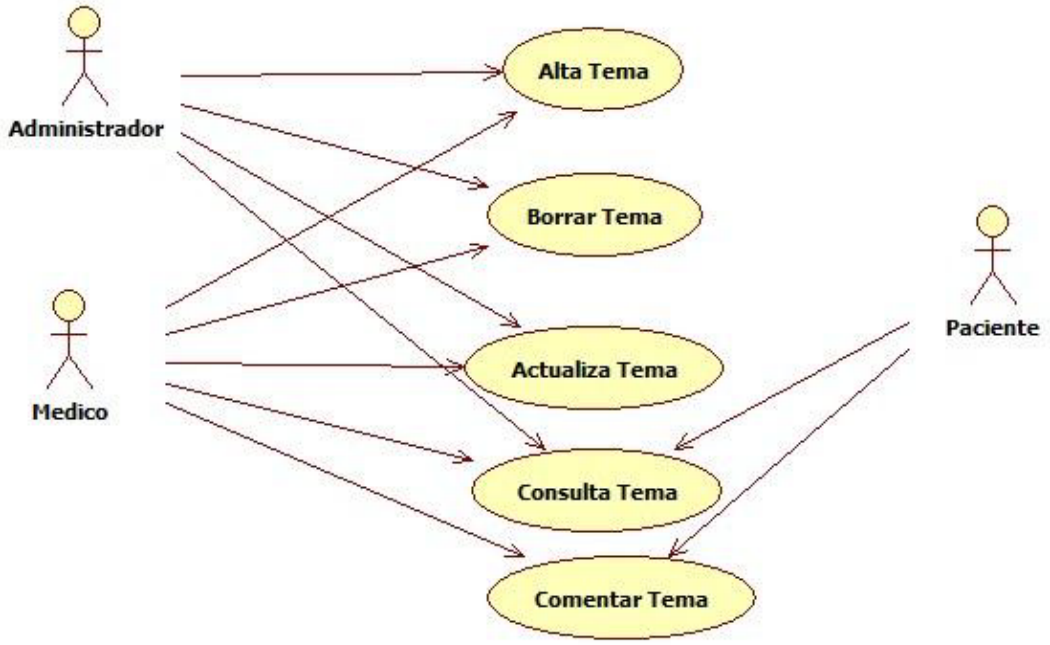


Caso de Uso:	Alta Respuesta
Descripción:	Modelar los escenarios que contemplan el registro de una respuesta en el sistema.
Actores:	Médico: Usuario que puede dar de alta una respuesta.
Precondiciones:	Que el Médico estén firmado en el sistema
Escenario Principal	El Médico ingresa a la forma para dar de alta una respuesta y llena el campo correspondiente.
Escenario alternativo:	Ninguno
Poscondiciones:	Se agrega una respuesta exitosamente.

Caso de Uso:	Borrar Respuesta
Descripción:	Modelar los escenarios que contemplan el borrado de una respuesta del sistema.
Actores:	Administrador, Médico. Administrador o Médico que desee borrar una respuesta en el sistema.
Precondiciones:	Debe existir la respuesta en el sistema.
Escenario Principal	El Administrador o Médico ingresa a la forma para dar de baja una respuesta y la borra de la lista correspondiente.
Escenario alternativo:	Ninguno
Poscondiciones:	Se borra una respuesta exitosamente.

Caso de Uso:	Consultar Respuesta
Descripción:	Modelar los escenarios que contemplan la consulta de una respuesta en el sistema.
Actores:	Administrador, Médico, Paciente. Administrador, Médico o Paciente que desee consultar una respuesta.
Precondiciones:	Debe existir la respuesta en el sistema.
Escenario Principal	El Administrador, Médico o Paciente ingresa a la forma para consultar una respuesta.
Escenario alternativo:	Ninguno
Poscondiciones:	Se consulta una respuesta exitosamente.

Temas



Caso de Uso:	Alta Tema
Descripción:	Modelar los escenarios que contemplan el registro de un tema al sistema
Actores:	Administrador, Médico. Usuario Administrador o Médico que puede dar de alta a un Tema.
Precondiciones:	Que el Administrador o Médico estén firmados en el sistema exitosamente.
Escenario Principal	El Administrador o Médico ingresan a la forma para dar de alta un Tema y llena los campos correspondientes.
Escenario alternativo:	Ninguno
Poscondiciones:	Se da de alta un Tema exitosamente

Caso de Uso:	Baja Tema
Descripción:	Modelar los escenarios que contemplan el borrado de un Tema en el sistema.
Actores:	Administrador, Médico. Administrador o Médico que desee borrar un Tema en el sistema.
Precondiciones:	Debe existir el Tema en el sistema.
Escenario Principal	El Administrador o Médico ingresan a la forma para dar de baja un Tema y borra el correspondiente.
Escenario alternativo:	Ninguno
Poscondiciones:	Se borra un Tema exitosamente.

Caso de Uso:	Actualiza Tema
Descripción:	Modelar los escenarios que contemplan la modificación de un Tema en el sistema.
Actores:	Administrador, Médico. Administrador o Médico que desee modificar la información de un Tema en el sistema.
Precondiciones:	Debe existir el Tema en el sistema.
Escenario Principal	El Administrador o Médico ingresan a la forma para modificar un Tema y actualiza los datos correspondientes.
Escenario alternativo:	Ninguno
Poscondiciones:	Se modifica un Tema exitosamente.

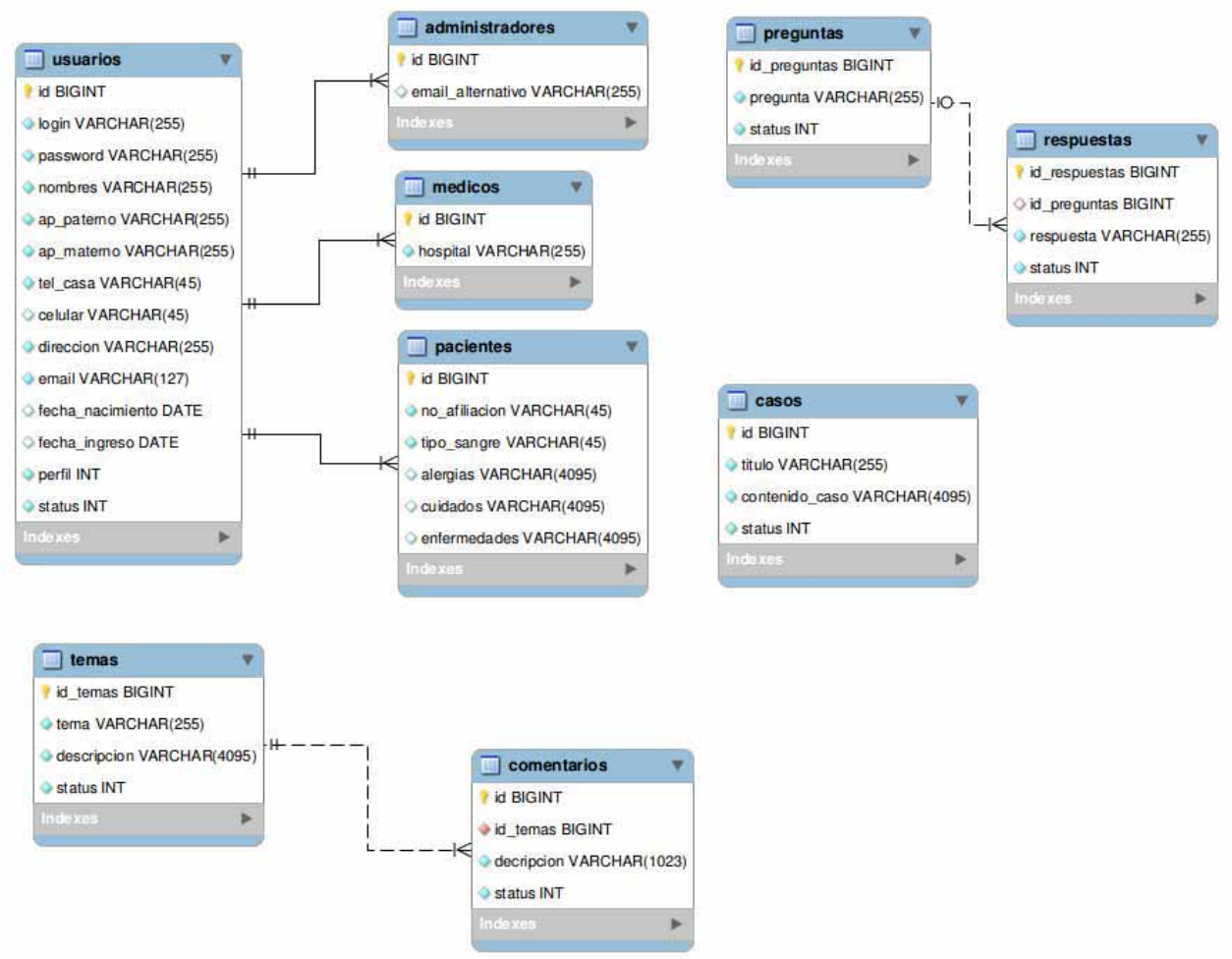
Caso de Uso:	Consultar Tema
Descripción:	Modelar los escenarios que contemplan la consulta de los datos de un Tema en el sistema.
Actores:	Médico, Paciente. Médico o Paciente que desee consultar la información de un Tema.
Precondiciones:	Debe existir el Tema en el sistema.
Escenario Principal	El Médico o Paciente ingresan a la forma para consultar la información de un Tema en el sistema.
Escenario alternativo:	Ninguno
Poscondiciones:	Se realiza una consulta de un Tema exitosamente.

Caso de Uso:	Comentar Tema
Descripción:	Modelar los escenarios que contemplan la facultad de agregar un comentario en un Tema del sistema.
Actores:	Médico, Paciente. Médico o Paciente que desee agregar un comentario en algún Tema del sistema.
Precondiciones:	Debe existir el Tema en el sistema.
Escenario Principal	El Médico o Paciente ingresan a la forma para agregar un comentario en el sistema.
Escenario alternativo:	Ninguno
Poscondiciones:	Se agrega exitosamente un comentario en un Tema en particular.

Implamentación del Sistema

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.



Diccionario de datos

Base de Datos ptForo

Tabla Usuarios. Contiene los campos que almacenan la información de los usuarios. Esta tabla se relaciona a su vez con las siguientes: Administradores, Médicos y Pacientes.

Campo id.- Es el número identificador único que se le asignara a cada registro de usuarios, administradores, médicos y pacientes.

Campo login.- Es el identificador único alfanumérico que los usuarios utilizarán para accesar al sistema.

Campo password.- Palabra clave que el usuario utilizará como contraseña para accesar al sistema.

Campo nombres.- Almacena el o los nombres de los usuarios.

Campo ap_paterno.- Almacena el apellido paterno de los usuarios.

Campo ap_materno.- Almacena el apellido materno de los usuarios.

Campo tel_casa.- Campo que guarda el teléfono de casa de los usuarios.

Campo celular.- Campo que guarda el número telefónico celular de los usuarios.

Campo dirección.- Almacena la dirección de los usuarios.

Campo email.- Guarda la dirección de correo electrónico de los usuarios.

Campo fecha_de_nacimiento.- Campo que almacena la fecha de nacimiento de cada usuario.

Campo fecha_de_ingreso.- Campo que guarda la fecha en que un paciente se interna en un hospital.

Campo perfil.- Almacena el tipo de usuario según corresponda. Puede ser administrador, médico o paciente.

Campo status.- Guarda el estado de cada usuario, pregunta, respuesta es decir, si es un usuario activo o inactivo en la base de datos.

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

Campo email_alternativo.- Campo que almacena un correo electrónico alterno para un administrador.

Tabla Médicos. Tabla destinada a almacenar la información de los médicos, en particular el hospital donde laboran los médicos.

Campo hospital.- Almacena el nombre del hospital en que labora un médico.

Tabla Pacientes. Contendrá los campos de los pacientes que guardaran información importante acerca de los pacientes.

Campo no_afiliación.- Campo que guarda el número con el que el hospital identifica al paciente.

Campo tipo_de_sangre.- Campo que almacena el tipo de sangre de cada paciente.

Campo cuidados.- Almacena las recomendaciones que debe seguir un paciente.

Campo enfermedades.- Guarda el registro de las enfermedades de cada paciente.

Tabla Preguntas. Contendrá los campos que guardaran las preguntas realizadas por los Pacientes.

Campo id_preguntas.- Número único que identifica a cada pregunta.

Campo pregunta.- Almacena como texto la pregunta que realiza el paciente.

Tabla Respuestas. Almacenará los campos que guardan la información de las respuestas generadas por los médicos.

Campo id_respuestas.- Número único de identificación para una respuesta asociado con una pregunta de un paciente.

Campo respuesta.- Almacena como texto la respuesta de un médico asociada a una pregunta de un paciente.

Tabla casos. Contendrá los campos de los casos generados por los médicos.

Campo id_casos.- Número identificador único que se le asigna a un texto que adjunta un médico con contenido de algún tema en particular.

Campo título.- Campo que almacena el título del texto que adjunta un médico con contenido de algún tema en particular.

Campo contenido_casos.- Almacena el texto del contenido informativo de algún tema que agrega alguno de los médicos.

Tabla Temas. Contiene los campos que almacenan información acerca de un tema agregado por un Médico.

Campo id_temas.- Almacena el número identificador único del tema que se tratará en el foro.

Campo tema.- Almacena el nombre del tema que se agrega en el foro.

Campo descripción_tema.- Campo que contiene una breve descripción del tema que se agrega al foro.

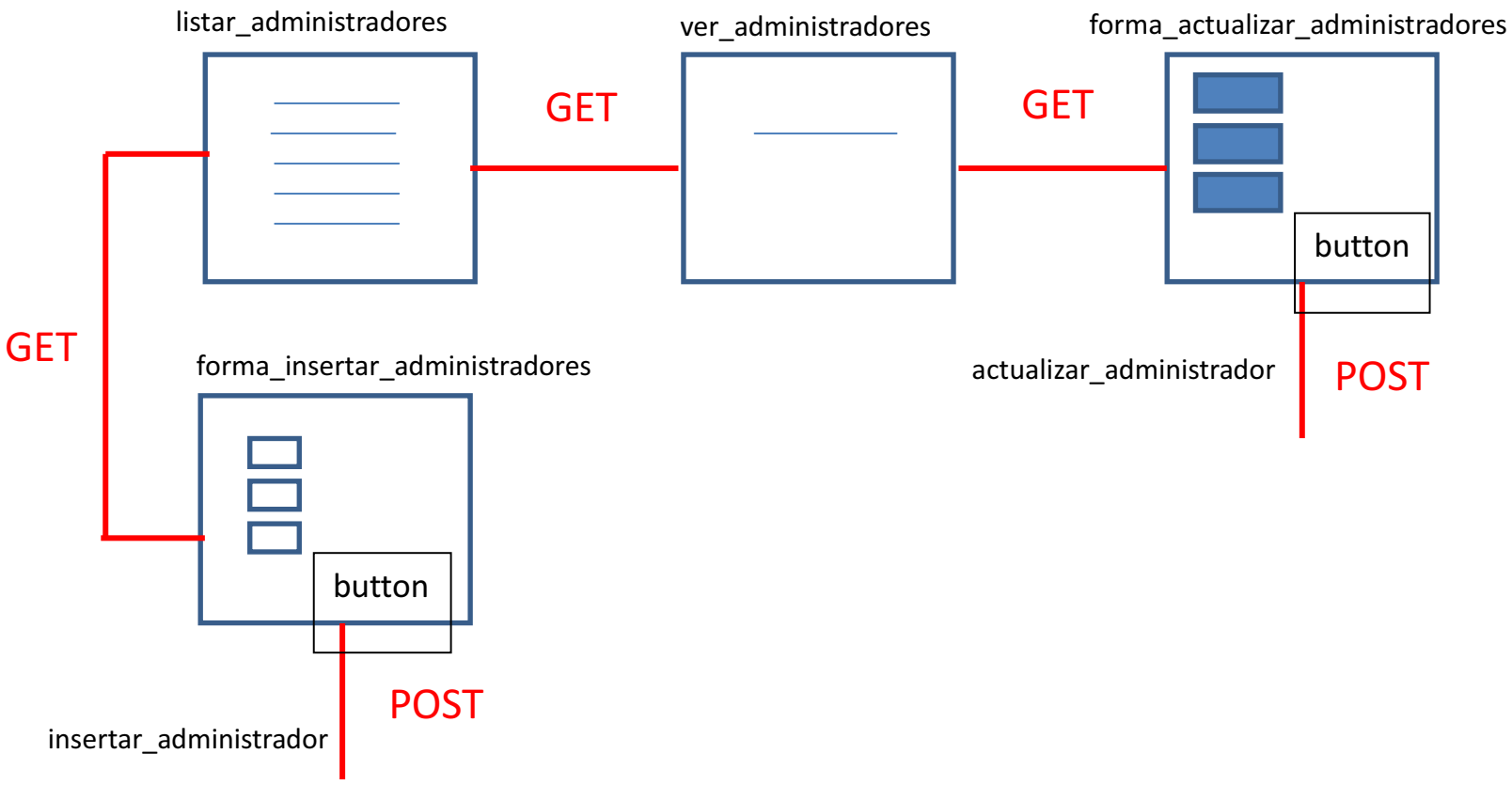
Tabla Comentarios: Contiene los campos que guardan la información de los comentarios agregados por los usuarios: Pacientes y Médicos.

Campo id_comentario.- Almacena el número identificador único del comentario que se hace sobre un tema en el foro.

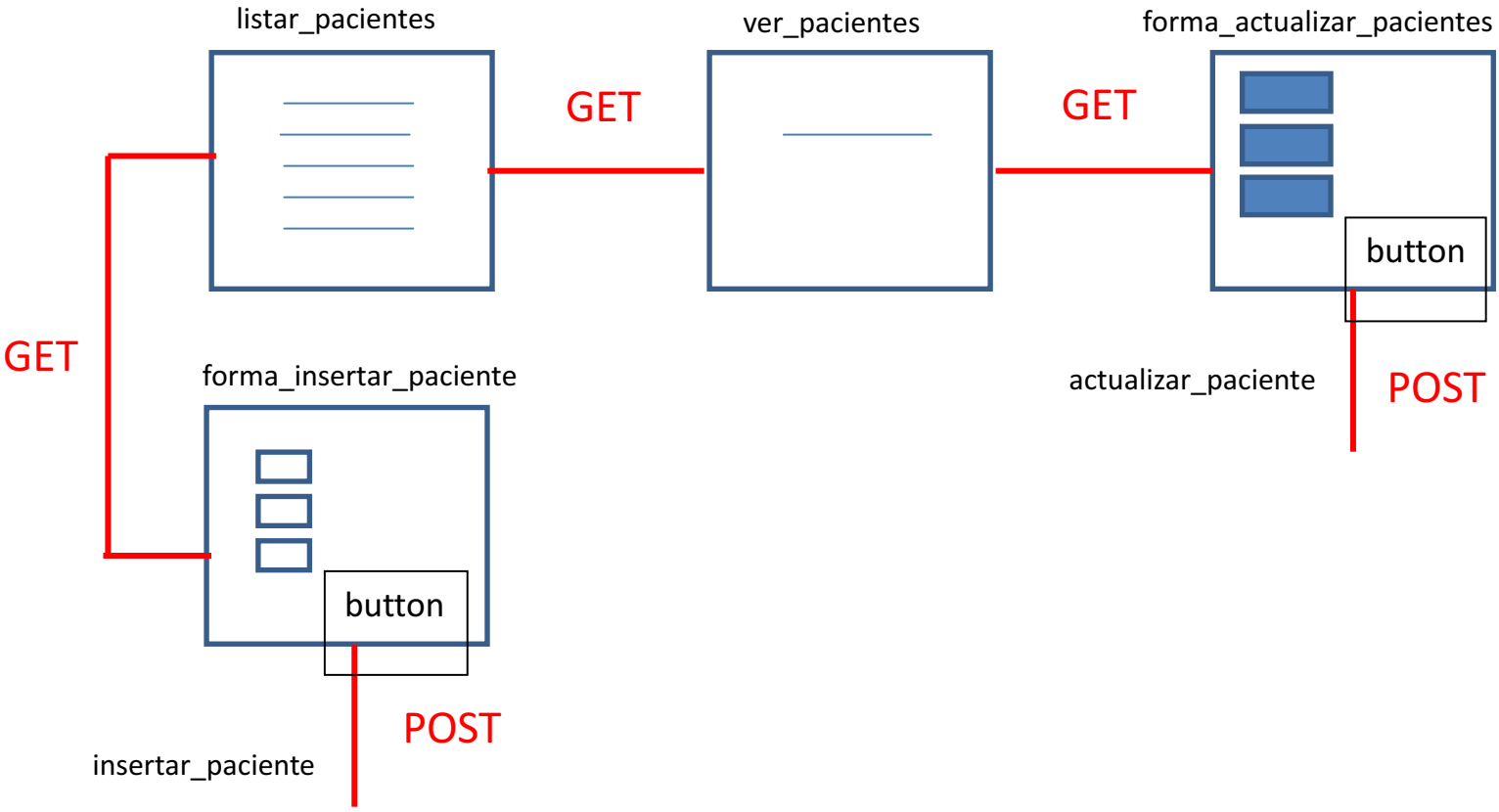
Campo descripcion_comentario.- Almacena el comentario que algún usuario hace en el foro.

Diagrama de Navegación entre páginas y clases

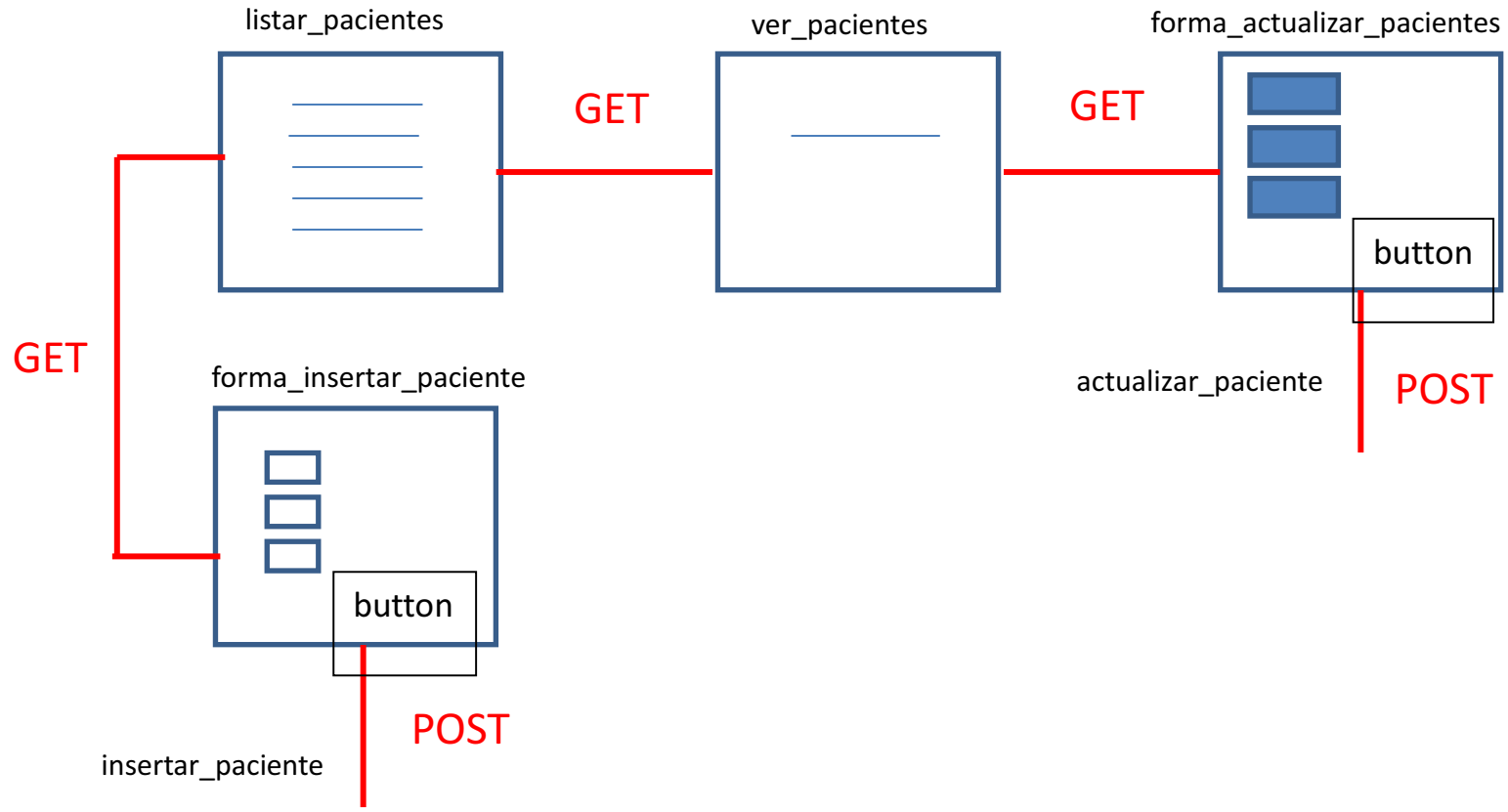
Administración Administradores



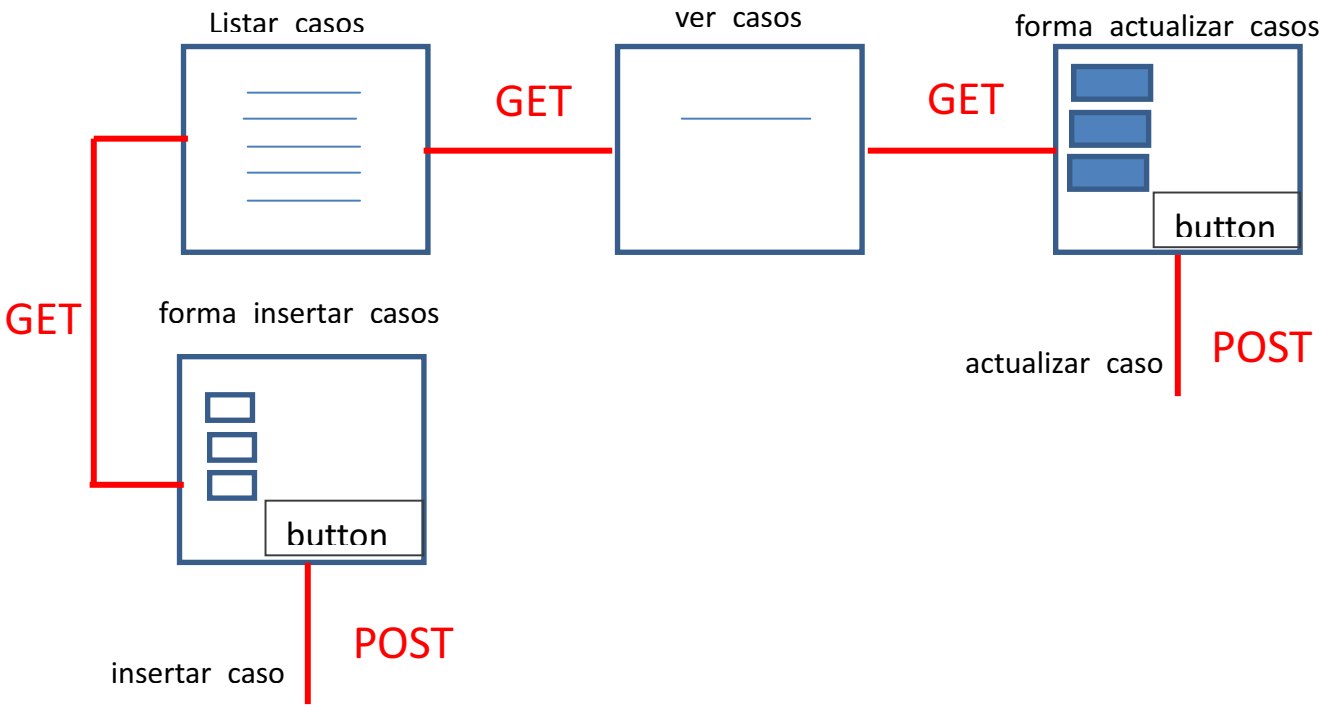
Administrador Pacientes



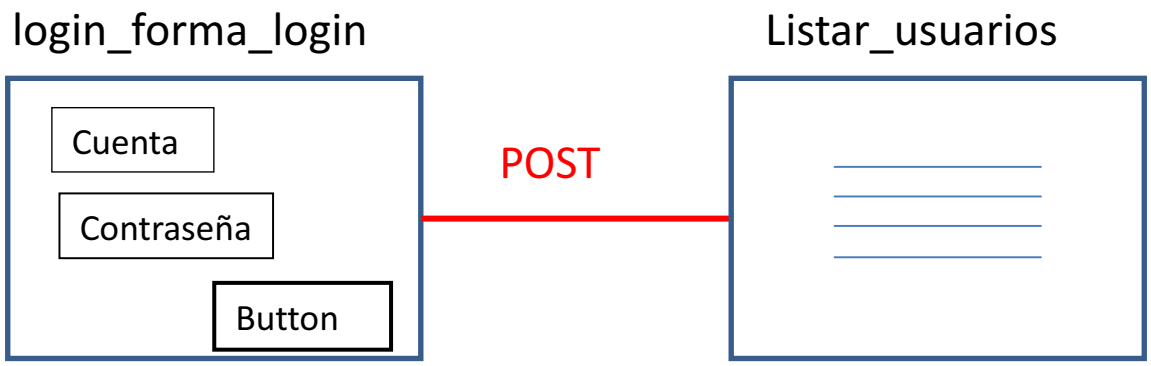
Administrador Médicos



Administración Casos



Administración Forma Login



Manual de Configuración

Para que el Sitema de Apoyo Emocional para Pacientes de Enfermedades Neoplásicas pueda funcionar se requiere la instalación en el sistema operativo Linux de los siguientes programas:

- jdk1.7.0_09
- apache-tomcat-6.0.37
- mySQL

Acontinuacion se detallará la instalación de cada uno de los programas anteriormente listados.

Jdk1.7.0_09

Lo primero que se debe hacer es entrar a la página de Oracle y buscar el archivo de instalación indicado. En el siguiente enlace se puede encontrar la versión que requerimos instalar http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html y manualmente aceptar el acuerdo de licencia. Esto debe hacerse desde una PC con una interfaz de usuario. Usted necesita obtener la versión correcta, que en mi caso era la versión tar.gz x64.

Ahora que se tiene el archivo en un servidor de Ubuntu se puede extraer usando:

tar-xvf jdk-7u9-linux-i586.tar.gz

Esto debe darle una "jdk1.7.0_09" directorio que tenemos que movernos a un lugar sensato como "/ usr/lib/jvm/jdk1.7.0_09":

sudo mv jdk1.7.0_09 / usr/lib/jvm/jdk1.7.0_09

Editamos el archivo /etc/profile/ para un usuario (el usuario actual)

Al final del archivo agregamos las siguientes líneas para poner valores a las varibles de ambiente.

```
JAVA_HOME="/usr/local/jdk1.7.0_09"

CLASSPATH="usr/local/mislibrerias"

PATH="$PATH:/usr/local/jdk1.7.0_09/bin"

export JAVA_HOME

export CLASSPATH

export PATH
```

y finalmente reiniciamos la PC

Una vez reiniciada la computadora comprobamos que Java está todo instalado correctamente comprobando la versión usando:

```
java-version
```

Que en este caso se debe dar:

```
java version "1.7.0_09"
Java (TM) Runtime Environment (build 1.7.0_09-b05)
Java HotSpot (TM) Client VM (build 23.5-b02, modo mixto)
```

apache-tomcat-6.0.37

Lo primero sera descargar el archive de la siguiente liga:

http://archive.apache.org/dist/tomcat/tomcat-6/v6.0.37/src/

Seleccionamos: Binary Distributions: tar.gz (pgp, md5)

```
Requisitos:
Privilegios de root
Tener instalado sun-java6-jdk y sun-java6-jre
```

Iniciamos:

Abrimos una Terminal

Nos logeamos como root:

su
passwd:
#

Copiamos el archivo descargado a /usr/local/

#cp /media/Datos/Software/Sistemas\ Linux/Servidores\ Web/Apache\ Tomcat/apache- tomcat-6.0.18.tar.gz /usr/local/

Descomprimimos y desempaquetamos:

#tar -zxvf apache-tomcat-6.0.18.tar.gz

Entramos a nuestro directorio home el mio es /home/rodrigo/

#cd /home/rodrigo

Ahora levantamos nuestro servidor Apache Tomcat

#/usr/local/apache-tomcat-6.0.18/bin/startup.sh

Nos muestran los siguientes mensajes:
Using CATALINA_BASE: /usr/local/apache-tomcat-6.0.18
Using CATALINA_HOME: /usr/local/apache-tomcat-6.0.18
Using CATALINA_TMPDIR: /usr/local/apache-tomcat-6.0.18/temp
Using JRE_HOME: /usr/lib/jvm/java-6-sun

Probamos en un navegador web:

http://localhost:8080/

Si nos aparece la página de Apache Tomcat, todo está correcto, ahora tendremos soporte para código de jsp y servlets de java.

Si deseamos parar nuestro servidor web:

#/usr/local/apache-tomcat-6.0.18/bin/shutdown.sh

El directorio de publicación es:

#cd /usr/local/apache-tomcat-6.0.18/webapps/PT_204359136

mysql

Primero tenemos que instalar mysql en ubuntu, para esto debemos instalar los paquetes mysql-server y mysql-client:

$sudo apt-get install mysql-server mysql-client

Una vez instalado modificamos la password de administrador de la siguiente forma:

$sudo /usr/bin/mysqladmin -u root password manager

de esta manera le indico el super usuario(root) y password(manager) que admistrará la base. Habiendo hecho esto nos conectamos como root de la siguiente forma:

$mysql -u root -p

Nos solicitará la password que en este caso es *manager*.

Manual de Usuario

Página principal

Aparecerá una pantalla con una forma en la cual se deben llenar los campos login y password, hay tres diferentes tipos de usuarios, dependiendo el tipo de usuario, se abrirá el menú correspondiente. Se deben llenar los campos Cuenta de Usuario y Contraseña y oprimir el botón enviar.

En caso de ingresar incorrectamente los datos, se redirigirá a la misma pantalla, para volver a intentarlo.

Para poder ingresar de manera correcta al sistema debe existir el usuario y la contraseña correspondiente para éste en la base de datos, es decir, previamente debe ser ingresado a la base, o manualmente en el sistema el usuario.

Existen 3 tipos de usuario: Administrador, Médico y Paciente.

Dependiendo del tipo de usuario que se ingrese será el menú que se muestre después de haber hecho un correcto ingreso con un usuario real existente en la base.

Usuario Administrador

Al ingresar un usuario tipo Administrador de manera exitosa se mostrará una lista con las siguientes opciones:

- Listar Médicos

- Listar Administradores

- Listar Pacientes

- Listar Casos

- Listar Preguntas

Listar Médicos

Al elegir la opción de Listar Médicos se mostrará una lista vacía y en el centro se desplegará un formulario de búsqueda en el cual al introducir un patrón que coincida con uno o más médicos los enlistará. Este patrón busca al administrador únicamente por su nombre o nombres.

Una vez listado el o los médicos que coincidieron con el patrón que se introdujo podremos ver el registro completo de dicho médico como se almacena en la base de datos. Si se desea mostrar toda la información de alguno de registros se debe dar click izquierdo en el número de ID que se muestra al principio de la lista.

Una vez desplegada toda la información de algún registro en la parte inferior se muestra la opción de Actualizar Médico.

Actualizar Médico

En el momento que se ha desplegado toda la información de un médico, tenemos la opción de actualizar dichos campos, para realizar alguna actualización debemos dar click en el botón de la parte inferior que dice: Actualizar Médico.

Cuando elegimos la opción de Actualizar Médico se muestra el contenido de cada campo del registro y se puede editar cada uno de estos a excepción del campo Perfil. Dada alguna modificación debemos volver a seleccionar el botón Actualizar Médico para que se guarden los cambios de lo contrario no se modificará el o los campos. Si por alguna razón ya no se desea modificar el registro se elige la opción Regresar.

Insertar Médico

Los Administradores también tienen la opción de Insertar un nuevo Médico. Para realizar la inserción de un médico se debe dar click izquierdo en el botón que se encuentra debajo del buscador de administradores que dice: Insertar Médico.

Cuando se ha elegido la opción Insertar Médico se despliega un formulario en el que se requiere introducir los datos del nuevo médico. Los campos requeridos para un médico son:

-Login

-Password

-Nombre(s)

-Apellido Paterno

-Apellido Materno

-Apellido Materno

-Celular

-Dirección

-Email

-Hospital

Cuando se han llenado los campos obligatorios ya se puede guardar el registro, para esto damos click en el botón Insertar Médico. Si por alguna razón se desea cancelar la inserción del nuevo médico debemos dar click en la opción Regresar.

Listar Administradores

Al elegir la opción de Listar Administradores se mostrará una lista vacía y en el centro se desplegará un formulario de busqueda en el cual al introducir un patrón que coincida con uno o más administradores los enlistará. Este patrón busca al administrador únicamente por su nombre o nombres.

Una vez listado el o los administradores que coincidieron con el patrón que se introdujo podremos ver el registro completo de dicho administrador como se almacena en la base de datos. Si se desea mostrar toda la información de alguno de registros se debe dar click izquierdo en el número de ID que se muestra al principio de la lista.

Una vez desplegada toda la información de algún registro en la parte inferior se muestra la opción de Actualizar Administrador.

Actualizar Administrador

En el momento que se ha desplegado toda la información de un administrador, tenemos la opción de actualizar dichos campos, para realizar alguna actualización debemos dar click en el botón de la parte inferior que dice: Actualizar Administrador.

Cuando elegimos la opción de Actualizar Administrador se muestra el contenido de cada campo del registro y se puede editar cada uno de estos a excepción del

campo Perfil. Dada alguna modificación debemos volver a seleccionar el botón Actualizar Administrador para que se guarden los cambios de lo contrario no se modificará el o los campos. Si por alguna razón ya no se desea modificar el registro se elige la opción Regresar.

Insertar Administrador

Los Administradores también tienen la opción de Insertar un nuevo Administrador. Para realizar la inserción de un administrador se debe dar click izquierdo en el botón que se encuentra debajo del buscador de administradores que dice: Insertar Administrador.

Cuando se ha elegido la opción Insertar Administrador se despliega un formulario en el que se requiere introducir los datos del nuevo administrador. Los campos requeridos para un administrador son:

-Login

-Password

-Nombre(s)

-Apellido Paterno

-Apellido Materno

-Apellido Materno

-Teléfono Casa

-Celular

-Dirección

-Email

-Email alternativo

Cuando se han llenado los campos obligatorios ya se puede guardar el registro, para esto damos click en el botón Insertar Administrador. Si por alguna razón se desea cancelar la inserción del nuevo administrador debemos dar click en la opción Regresar.

Listar Paciente

Al elegir la opción de Listar Paciente se mostrará una lista vacía y en el centro se desplegará un formulario de búsqueda en el cual al introducir un patrón que coincida con uno o más pacientes los enlistará. Este patrón busca al paciente únicamente por su nombre o nombres.

Una vez listado el o los pacientes que coincidieron con el patrón que se introdujo podremos ver el registro completo de dicho paciente como se almacena en la base de datos. Si se desea mostrar toda la información de alguno de registros se debe dar click izquierdo en el número de ID que se muestra al principio de la lista.

Una vez desplegada toda la información de algún registro en la parte inferior se muestra la opción de Actualizar Paciente.

Actualizar Paciente

En el momento que se ha desplegado toda la información de un paciente, tenemos la opción de actualizar dichos campos, para realizar alguna actualización debemos dar click en el boton de la parte inferior que dice: Actualizar Paciente.

Cuando elegimos la opción de Actualizar Paciente se muestra el contenido de cada campo del registro y se puede editar cada uno de estos a excepción del campo Perfil. Dada alguna modificación debemos volver a seleccionar el botón Actualizar Paciente para que se guarden los cambios de lo contrario no se modificará el o los campos. Si por alguna razón ya no se desea modificar el registro se elige la opción Regresar.

Insertar Paciente

Los Administradores también tienen la opción de Insertar un nuevo Paciente. Para realizar la inserción de un paciente se debe dar click izquierdo en el botón que se encuentra debajo del buscador de pacientes que dice: Insertar Paciente.

Nota: Para agregar un paciente el administrador previamente tuvo que haber elegido la opción Listar Pacientes.

Cuando se ha elegido la opción Insertar Paciente se despliega un formulario en el que se requiere introducir los datos del nuevo administrador. Los campos requeridos para un Paciente son:

-Login

-Password

-Nombre(s)

-Apellido Paterno

-Apellido Materno

-Apellido Materno

-Teléfono Casa

-Celular

-Dirección

-Email

-Número de afiliación

-Tipo de Sangre

-Alergias

-Cuidados

-Enfermedades

Cuando se han llenado los campos obligatorios ya se puede guardar el registro, para esto damos click en el botón Insertar Paciente. Si por alguna razón se desea cancelar la inserción del nuevo Paciente debemos dar click en la opción Regresar.

Listar casos

Cuando un médico ingresa al sistema puede dar de alta tema en particular (en nuestro sistema los llamamos Casos) e introducir información sobre éste, para esto debe elegir la opción Listar Casos.

Después se mostrará una lista vacía y en el centro se desplegará un formulario de búsqueda en el cual al introducir un patrón que coincida con alguno o varios de los casos los enlistará.

Una vez listado el o los casos que coincidieron con el patrón que se introdujo podremos ver el ID y el Titulo de cada caso. Cuando aparezca la lista de casos y se desee mostrar el contenido del caso se debe dar click izquierdo en el número de ID que se muestra al principio de la lista.

Actualizar Paciente

En el momento que se ha desplegado toda la información de un caso, tenemos la opción de actualizar dichos campos, para realizar alguna actualización debemos dar click en el botón de la parte inferior que dice: Actualizar Caso.

Cuando elegimos la opción de Actualizar Caso se muestra el contenido del caso seleccionado así como su respectivo Título. Dada alguna modificación debemos volver a seleccionar el botón Actualizar Caso para que se guarden los cambios de

lo contrario no se modificará el o los campos. Si por alguna razón ya no se desea modificar el registro se elige la opción Regresar.

Insertar Caso

Los Administradores o Médicos también tienen la opción de Insertar un nuevo Caso. Para realizar la inserción de un caso se debe dar click izquierdo en el botón que se encuentra debajo del buscador de Casos que dice: Insertar Caso.

Nota: Para agregar un Caso el administrador o Médico previamente tuvo que haber elegido la opción Listar Pacientes.

Cuando se ha elegido la opción Insertar Caso se despliega un formulario en el que se requiere introducir los datos del nuevo Caso. Los campos requeridos para un Caso son:

-Título

-Contenido

Cuando se han llenado los campos obligatorios ya se puede guardar el registro, para esto damos click en el botón Insertar Caso. Si por alguna razón se desea cancelar la inserción del nuevo Caso debemos dar click en la opción Regresar.

Codigo fuente

InicializadorCatalogos.java

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

import java.util.Map;
import java.util.TreeMap;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

/**
 * Application Lifecycle Listener implementation class InicializadorCatalogos
 *
 */
public class InicializadorCatalogos implements ServletContextListener {
```

```
	public void contextInitialized(ServletContextEvent event) {
		ServletContext context = event.getServletContext();
		cargarTipoAudiencias( context );
	}

	public void contextDestroyed(ServletContextEvent event) {
	}

	private void cargarTipoAudiencias( ServletContext context )
	{
	Map<Character,String> audiencias = new TreeMap<Character,String>();
	audiencias.put( 'm', "Medicos" );
	audiencias.put( 'p', "Pacientes" );
	audiencias.put( 'a', "Administradores" );
	context.setAttribute( "audiencias", audiencias );
	}

}
```

AdministradorAction.java

```
package mx.uam.azc.pt.actions;

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

import mx.uam.azc.pt.business.AdministradorAdministradores;
import mx.uam.azc.pt.business.AdministradorCasos;
import mx.uam.azc.pt.business.AdministradorMedicos;
import mx.uam.azc.pt.business.AdministradorPacientes;
import mx.uam.azc.pt.business.AdministradorPreguntas;
import mx.uam.azc.pt.business.local.FiltroAdministradores;
import mx.uam.azc.pt.business.local.FiltroCasos;
import mx.uam.azc.pt.business.local.FiltroMedicos;
import mx.uam.azc.pt.business.local.FiltroPacientes;
import mx.uam.azc.pt.business.local.FiltroPreguntas;
import mx.uam.azc.pt.data.AdministradorJoinDTO;
import mx.uam.azc.pt.data.AdministradorSaveDTO;
import mx.uam.azc.pt.data.CasoJoinDTO;
import mx.uam.azc.pt.data.CasoSaveDTO;
import mx.uam.azc.pt.data.MedicoJoinDTO;
import mx.uam.azc.pt.data.MedicoSaveDTO;
import mx.uam.azc.pt.data.PacienteJoinDTO;
import mx.uam.azc.pt.data.PacienteSaveDTO;
import mx.uam.azc.pt.data.PreguntaJoinDTO;
import mx.uam.azc.pt.data.PreguntaSaveDTO;

import org.apache.struts2.ServletActionContext;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts2.interceptor.SessionAware;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;

/**
 * Clase AdministradorAction Maestra, funge como clase Action principal para todo el manejo
```

```
 * de flujos en struts
 * @author Rodrigo Gonzalez
 *
 */
public class AdministradorAction extends ActionSupport implements SessionAware,
                Preparable {

        private AdministradorAdministradores administradorAdministradores;
        private AdministradorPacientes administradorPacientes;
        private AdministradorCasos administradorCasos;
        private AdministradorPreguntas administradorPreguntas;
        private AdministradorMedicos administradorMedicos;
        private AdministradorJoinDTO administrador;
        private AdministradorSaveDTO administrador_save;
        private PacienteJoinDTO paciente;
        private PacienteSaveDTO paciente_save;
        private MedicoJoinDTO medico;
        private MedicoSaveDTO medico_save;
        private CasoJoinDTO caso;
        private CasoSaveDTO caso_save;
        private PreguntaJoinDTO pregunta;
        private PreguntaSaveDTO pregunta_save;
        private List<AdministradorJoinDTO> administradores = new
ArrayList<AdministradorJoinDTO>();
        private List<PacienteJoinDTO> pacientes = new ArrayList<PacienteJoinDTO>();
        private List<CasoJoinDTO> casos = new ArrayList<CasoJoinDTO>();
        private List<PreguntaJoinDTO> preguntas = new ArrayList<PreguntaJoinDTO>();
        private List<MedicoJoinDTO> medicos = new ArrayList<MedicoJoinDTO>();
        public List<MedicoJoinDTO> getMedicos() {
                return medicos;
        }

        public void setMedicos(List<MedicoJoinDTO> medicos) {
                this.medicos = medicos;
        }

        private long id;
        private long idMa;
        private long idA;
        private long idP;
        private long idC;
        private long idCn;
        private long idEx;
        /**
         * Variable para el manejo de sesion con struts
         */
        private Map<String, Long> sessionData;
        private FiltroAdministradores filtroad = new FiltroAdministradores();
        private FiltroPacientes filtrop = new FiltroPacientes();
        private FiltroCasos filtroc = new FiltroCasos();
        private FiltroMedicos filtrom = new FiltroMedicos();
        private FiltroPreguntas filtropr = new FiltroPreguntas();

        public FiltroMedicos getFiltrom() {
                return filtrom;
        }
```

```
public void setFiltro(FiltroMedicos filtrom) {
        this.filtrom = filtrom;
}

/**
 * Metodo que lee un listado de administradores acorde un filtro
 * @return Redireccion hacia el listado de administradores
 */
public String listar_administradores() {
        sessionData.put("idAdministrador", null);
        //filtroad.set_nombres("");
        System.out.println("Filtro: " + filtroad.getNombres());
        administradores = administradorAdministradores.leerAdministradoresFiltro(filtroad);
        System.out.println("Tamanio" + administradores.size());
        addActionMessage( "La busqueda Administrador fue exitosa" );
        return "listar_administradores";
}

/**
 * Metodo que lee un listado de medicos acorde un filtro
 * @return Redireccion hacia el listado de medicos
 */
public String listar_medicos() {
        sessionData.put("idMedico", null);
        //filtro.setNombres("juan");
        //System.out.println(filtro.getNombres());
        System.out.println("Filtro: " + filtrom.getNombres());
        medicos = administradorMedicos.leerMedicosFiltro(filtrom);
        System.out.println("Tamanio" + medicos.size());
        addActionMessage( "La busqueda Medicos fue exitosa" );
        return "listar_medicos";
}

/**
 * Metodo que lee a un administrador acorde a un id dado
 * @return La redireccion hacie la vista del administrador
 */
public String ver_administradores() {
        if(id == 0)
                id = ((Long)sessionData.get( "idAdministrador")).longValue();
        administrador = administradorAdministradores
                        .leerAdministrador(id);
        sessionData.put("idAdministrador", id);
        System.out.println("ver administradores: "
                        + administrador.getId());
        return "ver_administradores";
}

/**
 * Metodo que solo redireccion hacia la forma de insercion de administradores
 * @return La redirecion hacia la forma de insercion de administradores
 */
public String forma_insertar_administradores() {
        return "forma_insertar_administradores";
}
```

```
	/**
	 * Metodo que Inserta a un administrador a la BD y redireccion hacia el listado de
administradores
	 * @return La redirecion hacia el listado de administradores
	 */
	public String insertar_administradores() {
		administrador.setPerfil(1);
		administradorAdministradores.insertarAdministrador(administrador);
		addActionMessage( "El Administrador ha sido insertado" );
		return "listar_administradores_redirect";
	}

	/**
	 * Metodo que Inserta a un medico a la BD y redireccion hacia el listado de medicos
	 * @return La redirecion hacia el listado de medicos
	 */
	public String insertar_medicos() {
		medico.setPerfil(2);
		administradorMedicos.insertarMedico(medico);
		addActionMessage( "El medico ha sido insertado" );
		return "listar_medicos_redirect";
	}

	/**
	 * Metodo que redirecciona a la forma de actualizacion de un administrador
	 * @return La redireccion hacia la forma de actualizacion con los datos precargados
	 */
	public String forma_actualizar_administradores() {
		administrador = administradorAdministradores
				.leerAdministrador(administrador.getId());
		administrador_save = cargarAdministrador(administrador);
		return "forma_actualizar_administradores";
	}

	/**
	 * Metodo que actualiza a un administrador ya percargado en la sesion
	 * @return La redireccion hacia el listado de administradores
	 */
	public String actualizar_administradores() {

		administradorAdministradores.actualizarAdministrador(administrador);
		addActionMessage( "El Administrador ha sido actualizado" );
		sessionData.put( "idAdministrador", administrador.getId() );
		return "listar_administradores_redirect";
	}

	/**
	 * Metodo que deshabilita a un administrador, cambia su status a 1
	 * @return La redireccion hacia el listado de administradores
	 */
	public String borrar_administradores() {
		administrador = administradorAdministradores
				.leerAdministrador(id);
		administrador.setStatus(1);
		administradorAdministradores.actualizarAdministrador(administrador);
```

```
                return "listar_administradores_redirect";
        }

        /**
         * Metodo que redireccion al menu de administrador
         * @return La redireccin hacie el menu de administrador y/o medico adscrito
         */
        public String menu() {
                return "menu";
        }
        /**
         * Metodo que carga un administrador con algun tipo de relacion a un administrador sin
relacion alguna
         * @param administrador Objeto con algun tipo de relacion
         * @return El administrador sin relacion alguna
         */
        private AdministradorSaveDTO cargarAdministrador(
                        AdministradorJoinDTO administrador) {
                AdministradorSaveDTO ad = new AdministradorSaveDTO();
                ad.setAp_materno(administrador.getAp_materno());
                ad.setAp_paterno(administrador.getAp_paterno());
                ad.setCelular(administrador.getCelular());
                ad.setDireccion(administrador.getDireccion());
                ad.setEmail(administrador.getEmail());
                ad.setEmail_alternativo(administrador.getEmail_alternativo());
                ad.setFecha_ingreso(administrador.getFecha_ingreso());
                ad.setFecha_nacimiento(administrador.getFecha_nacimiento());
                ad.setId(administrador.getId());
                ad.setLogin(administrador.getLogin());
                ad.setNombres(administrador.getNombres());
                ad.setPassword(administrador.getPassword());
                ad.setPerfil(administrador.getPerfil());
                ad.setStatus(administrador.getStatus());
                ad.setTel_casa(administrador.getTel_casa());
                return ad;
        }

        /**
         * Metodo que lee un listado de pacientes acorde un filtro
         * @return Redireccion hacia el listado de pacientes
         */
        public String listar_pacientes() {
                sessionData.put("idPaciente", null);
                //filtroad.set_nombres("");
                pacientes = administradorPacientes.leerPacientesFiltro(filtrop);
                System.out.println("Tamanio" + pacientes.size());
                addActionMessage( "La busqueda Paciente fue exitosa" );
                return "listar_pacientes";
        }


        /**
         * Metodo que lee un listado de casos acorde un filtro
         * @return Redireccion hacia el listado de casos
         */
        public String listar_casos() {
```

```
		sessionData.put("idCaso", null);
		casos = administradorCasos.leerCasosFiltro(filtroc);
		System.out.println("Tamanio" + casos.size());
		addActionMessage( "La busqueda Caso fue exitosa" );
		return "listar_casos";
	}

	/**
	 * Metodo que lee un listado de preguntas acorde un filtro
	 * @return Redireccion hacia el listado de preguntas
	 */
	public String listar_preguntas() {
		sessionData.put("idPregunta", null);
		preguntas = administradorPreguntas.leerPreguntasFiltro(filtropr);
		System.out.println("Tamanio" + preguntas.size());
		addActionMessage( "La busqueda de la pregunta fue exitosa" );
		return "listar_preguntas";
	}

	/**
	 * Metodo que lee a un paciente acorde a un id dado
	 * @return La redireccion hacie la vista del paciente
	 */
	public String ver_pacientes() {
		if(id == 0)
			id = ((Long)sessionData.get( "idPaciente")).longValue();
		paciente = administradorPacientes
				.leerPaciente(id);
		sessionData.put("idPaciente", id);
		System.out.println("ver pacientes: "
				+ paciente.getId());
		return "ver_pacientes";
	}

	/**
	 * Metodo que lee a un medico acorde a un id dado
	 * @return La redireccion hacie la vista del medico
	 */
	public String ver_medicos() {
		if(id == 0)
			id = ((Long)sessionData.get( "idMedico")).longValue();
		medico = administradorMedicos
				.leerMedico(id);
		sessionData.put("idMedico", id);
		System.out.println("ver medicos: "
				+ medico.getId());
		return "ver_medicos";
	}

	/**
	 * Metodo que lee a un caso acorde a un id dado
	 * @return La redireccion hacie la vista del caso
	 */
	public String ver_casos() {
		if(id == 0)
			id = ((Long)sessionData.get( "idCaso")).longValue();
```

```
            caso = administradorCasos
                                .leerCaso(id);
            sessionData.put("idCaso", id);
            System.out.println("ver casos: "
                                + caso.getId());
            return "ver_casos";
    }

    /**
     * Metodo que solo redireccion hacia la forma de insercion de medicos
     * @return La redirecion hacia la forma de insercion de medicos
     */
    public String forma_insertar_medicos() {
            return "forma_insertar_medicos";
    }

    /**
     * Metodo que solo redireccion hacia la forma de insercion de pacientes
     * @return La redirecion hacia la forma de insercion de pacientes
     */
    public String forma_insertar_pacientes() {
            return "forma_insertar_pacientes";
    }

    /**
     * Metodo que solo redireccion hacia la forma de insercion de casos
     * @return La redirecion hacia la forma de insercion de casos
     */
    public String forma_insertar_casos() {
            return "forma_insertar_casos";
    }

    /**
     * Metodo que Inserta a un paciente a la BD y redireccion hacia el listado de pacientes
     * @return La redirecion hacia el listado de pacientes
     */
    public String insertar_pacientes() {
            paciente.setPerfil(3);
            administradorPacientes.insertarPaciente(paciente);
            addActionMessage( "El Paciente ha sido insertado" );
            return "listar_pacientes_redirect";
    }

    /**
     * Metodo que Inserta a un caso a la BD y redireccion hacia el listado de casos
     * @return La redirecion hacia el listado de casos
     */
    public String insertar_casos() {
            administradorCasos.insertarCaso(caso);
            addActionMessage( "El Caso ha sido insertado" );
            return "listar_casos_redirect";
    }

    /**
     * Metodo que redirecciona a la forma de actualizacion de un paciente
     * @return La redireccion hacia la forma de actualizacion con los datos precargados
```

```
 */
public String forma_actualizar_pacientes() {
        paciente = administradorPacientes
                        .leerPaciente(paciente.getId());
        paciente_save = cargarPaciente(paciente);
        return "forma_actualizar_pacientes";
}

/**
 * Metodo que redirecciona a la forma de actualizacion de un medico
 * @return La redireccion hacia la forma de actualizacion con los datos precargados
 */
public String forma_actualizar_medicos() {
        medico = administradorMedicos
                        .leerMedico(medico.getId());
        medico_save = cargarMedico(medico);
        return "forma_actualizar_medicos";
}

/**
 * Metodo que redirecciona a la forma de actualizacion de un caso
 * @return La redireccion hacia la forma de actualizacion con los datos precargados
 */
public String forma_actualizar_casos() {
        System.out.println("caso No: " + caso.getId());
        caso = administradorCasos
                        .leerCaso(caso.getId());
        System.out.println("caso No: " + caso.getId());
        caso_save = cargarCaso(caso);
        return "forma_actualizar_casos";
}

/**
 * Metodo que actualiza a un paciente ya percargado en la sesion
 * @return La redireccion hacia el listado de pacientes
 */
public String actualizar_pacientes() {

        administradorPacientes.actualizacionPaciente(paciente);
        addActionMessage( "El Paciente ha sido actualizado" );
        sessionData.put( "idPaciente", paciente.getId() );
        return "listar_pacientes_redirect";
}

/**
 * Metodo que actualiza a un paciente ya percargado en la sesion
 * @return La redireccion hacia el listado de pacientes
 */
public String actualizar_medicos() {

        administradorMedicos.actualizarMedico(medico);
        addActionMessage( "El medico ha sido actualizado" );
        sessionData.put( "idMedico", medico.getId() );
        return "listar_medicos_redirect";
}
```

```
/**
 * Metodo que actualiza a un caso ya percargado en la sesion
 * @return La redireccion hacia el listado de casos
 */
public String actualizar_casos() {

        administradorCasos.actualizarCaso(caso);
        addActionMessage( "El Caso ha sido actualizado" );
        sessionData.put( "idCaso", caso.getId() );
        return "listar_casos_redirect";
}

/**
 * Metodo que deshabilita a un paciente, cambia su status a 1
 * @return La redireccion hacia el listado de pacientes
 */
public String borrar_pacientes() {
        paciente = administradorPacientes
                        .leerPaciente(id);
        paciente.setStatus(1);
        administradorPacientes.actualizacionPaciente(paciente);
        return "listar_pacientes_redirect";
}

/**
 * Metodo que deshabilita a un paciente, cambia su status a 1
 * @return La redireccion hacia el listado de pacientes
 */
public String borrar_medicos() {
        medico = administradorMedicos
                        .leerMedico(id);
        medico.setStatus(1);
        administradorMedicos.actualizarMedico(medico);
        return "listar_medicos_redirect";
}

/**
 * Metodo que deshabilita a un caso, cambia su status a 1
 * @return La redireccion hacia el listado de casos
 */
public String borrar_casos() {
        caso = administradorCasos
                        .leerCaso(id);
        caso.setStatus(1);
        administradorCasos.actualizarCaso(caso);
        return "listar_casos_redirect";
}

/**
 * Metodo que carga un medico  con algun tipo de relacion a un medico  sin relacion alguna
 * @param medico Objeto con algun tipo de relacion
 * @return El medico  sin relacion alguna
 */
private MedicoSaveDTO cargarMedico(
                MedicoJoinDTO medico) {
        MedicoSaveDTO m = new MedicoSaveDTO();
```

```
                m.setAp_materno(medico.getAp_materno());
                m.setAp_paterno(medico.getAp_paterno());
                m.setCelular(medico.getCelular());
                m.setDireccion(medico.getDireccion());
                m.setEmail(medico.getEmail());
                m.setFecha_ingreso(medico.getFecha_ingreso());
                m.setFecha_nacimiento(medico.getFecha_nacimiento());
                m.setId(medico.getId());
                m.setLogin(medico.getLogin());
                m.setNombres(medico.getNombres());
                m.setPassword(medico.getPassword());
                m.setPerfil(medico.getPerfil());
                m.setStatus(medico.getStatus());
                m.setTel_casa(medico.getTel_casa());
                return m;
        }

        /**
         * Metodo que carga un paciente con algun tipo de relacion a un paciente sin relacion
alguna
         * @param paciente Objeto con algun tipo de relacion
         * @return El paciente sin relacion alguna
         */
        private PacienteSaveDTO cargarPaciente(
                        PacienteJoinDTO paciente) {
                PacienteSaveDTO pac = new PacienteSaveDTO();
                pac.setAp_materno(paciente.getAp_materno());
                pac.setAp_paterno(paciente.getAp_paterno());
                pac.setCelular(paciente.getCelular());
                pac.setDireccion(paciente.getDireccion());
                pac.setEmail(paciente.getEmail());
                pac.setAlergias(paciente.getAlergias());
                pac.setCuidados(paciente.getCuidados());
                pac.setEnfermedades(paciente.getEnfermedades());
                pac.setNo_afiliacion(paciente.getNo_afiliacion());
                pac.setTipo_sangre(paciente.getTipo_sangre());
                pac.setFecha_ingreso(paciente.getFecha_ingreso());
                pac.setFecha_nacimiento(paciente.getFecha_nacimiento());
                pac.setId(paciente.getId());
                pac.setLogin(paciente.getLogin());
                pac.setNombres(paciente.getNombres());
                pac.setPassword(paciente.getPassword());
                pac.setPerfil(paciente.getPerfil());
                pac.setStatus(paciente.getStatus());
                pac.setTel_casa(paciente.getTel_casa());
                return pac;
        }

        /**
         * Metodo que carga un caso con algun tipo de relacion a un caso sin relacion alguna
         * @param caso Objeto con algun tipo de relacion
         * @return El caso sin relacion alguna
         */
        private CasoSaveDTO cargarCaso(
                        CasoJoinDTO caso) {
                CasoSaveDTO ca = new CasoSaveDTO();
```

```
            ca.setId(caso.getId());
            ca.setContenido_caso(caso.getContenido_caso());
            ca.setStatus(caso.getStatus());
            ca.setTitulo(caso.getTitulo());
            return ca;
}

/**
 * @return the administradorAdministradores
 */
public AdministradorAdministradores getAdministradorAdministradores() {
            return administradorAdministradores;
}

/**
 * @param administradorAdministradores
 *            the administradorAdministradores to set
 */
public void setAdministradorAdministradores(
                        AdministradorAdministradores administradorAdministradores) {
            this.administradorAdministradores = administradorAdministradores;
}
/**
 * @return the id
 */
public long getId() {
            return id;
}

/**
 * @param id
 * the id to set
 */
public void setId(long id) {
            this.id = id;
}

/**
 * @return the sessionData
 */
public Map getSessionData() {
            return sessionData;
}

/**
 * @param sessionData
 *            the sessionData to set
 */
public void setSessionData(Map sessionData) {
            this.sessionData = sessionData;
}

public void setSession(Map sessionData) {
            this.sessionData = sessionData;

}
```

```java
/**
 * @return the idMa
 */
public long getIdMa() {
	return idMa;
}

/**
 * @param idMa the idMa to set
 */
public void setIdMa(long idMa) {
	this.idMa = idMa;
}

/**
 * @return the administrador
 */
public AdministradorJoinDTO getAdministrador() {
	return administrador;
}

/**
 * @param administrador the administrador to set
 */
public void setAdministrador(AdministradorJoinDTO administrador) {
	this.administrador = administrador;
}

/**
 * @return the administrador_save
 */
public AdministradorSaveDTO getAdministrador_save() {
	return administrador_save;
}

/**
 * @param administrador_save the administrador_save to set
 */
public void setAdministrador_save(AdministradorSaveDTO administrador_save) {
	this.administrador_save = administrador_save;
}

/**
 * @return the administradores
 */
public List<AdministradorJoinDTO> getAdministradores() {
	return administradores;
}

/**
 * @param administradores the administradores to set
 */
public void setAdministradores(List<AdministradorJoinDTO> administradores) {
	this.administradores = administradores;
}
```

```
/**
 * @return the idA
 */
public long getIdA() {
	return idA;
}

/**
 * @param idA the idA to set
 */
public void setIdA(long idA) {
	this.idA = idA;
}

/**
 * @return the filtroad
 */
public FiltroAdministradores getFiltroad() {
	return filtroad;
}

/**
 * @param filtroad the filtroad to set
 */
public void setFiltroad(FiltroAdministradores filtroad) {
	this.filtroad = filtroad;
}

/**
 * @return the idP
 */
public long getIdP() {
	return idP;
}

/**
 * @param idP the idP to set
 */
public void setIdP(long idP) {
	this.idP = idP;
}

/**
 * @return the filtrop
 */
public FiltroPacientes getFiltrop() {
	return filtrop;
}

/**
 * @param filtrop the filtrop to set
 */
public void setFiltrop(FiltroPacientes filtrop) {
	this.filtrop = filtrop;
}
```

```
/**
 * @return the idC
 */
public long getIdC() {
	return idC;
}

/**
 * @param idC the idC to set
 */
public void setIdC(long idC) {
	this.idC = idC;
}

/**
 * @return the filtroc
 */
public FiltroCasos getFiltroc() {
	return filtroc;
}

/**
 * @param filtropr the filtropr to set
 */
public void setFiltropr(FiltroPreguntas filtropr) {
	this.filtropr = filtropr;
}

/**
 * @return the filtropr
 */
public FiltroPreguntas getFiltropr() {
	return filtropr;
}

/**
 * @param filtroc the filtroc to set
 */
public void setFiltroc(FiltroCasos filtroc) {
	this.filtroc = filtroc;
}

/**
 * @return the idCn
 */
public long getIdCn() {
	return idCn;
}

/**
 * @param idCn the idCn to set
 */
public void setIdCn(long idCn) {
	this.idCn = idCn;
}
```

```
//@Override
public void prepare() throws Exception {
	// TODO Auto-generated method stub

}

public AdministradorPacientes getAdministradorPacientes() {
	return administradorPacientes;
}

public void setAdministradorPacientes(AdministradorPacientes administradorPacientes) {
	this.administradorPacientes = administradorPacientes;
}

public AdministradorCasos getAdministradorCasos() {
	return administradorCasos;
}

public void setAdministradorCasos(AdministradorCasos administradorCasos) {
	this.administradorCasos = administradorCasos;
}

public AdministradorPreguntas getAdministradorPreguntas() {
	return administradorPreguntas;
}

public void setAdministradorPreguntas(AdministradorPreguntas administradorPreguntas) {
	this.administradorPreguntas = administradorPreguntas;
}

//
public AdministradorMedicos getAdministradorMedicos() {
	return administradorMedicos;
}

public void setAdministradorMedicos(AdministradorMedicos administradorMedicos) {
	this.administradorMedicos = administradorMedicos;
}
//

public PacienteJoinDTO getPaciente() {
	return paciente;
}

public void setPaciente(PacienteJoinDTO paciente) {
	this.paciente = paciente;
}

public PacienteSaveDTO getPaciente_save() {
	return paciente_save;
}

public void setPaciente_save(PacienteSaveDTO paciente_save) {
	this.paciente_save = paciente_save;
}
```

```
public CasoJoinDTO getCaso() {
	return caso;
}

public void setCaso(CasoJoinDTO caso) {
	this.caso = caso;
}

public CasoSaveDTO getCaso_save() {
	return caso_save;
}

public void setCaso_save(CasoSaveDTO caso_save) {
	this.caso_save = caso_save;
}

public PreguntaJoinDTO getPregunta() {
	return pregunta;
}

public void setPregunta(PreguntaJoinDTO pregunta) {
	this.pregunta = pregunta;
}

public PreguntaSaveDTO getPregunta_save() {
	return pregunta_save;
}

public void setPregunta_save(PreguntaSaveDTO pregunta_save) {
	this.pregunta_save = pregunta_save;
}

public List<PacienteJoinDTO> getPacientes() {
	return pacientes;
}

public void setPacientes(List<PacienteJoinDTO> pacientes) {
	this.pacientes = pacientes;
}

public List<CasoJoinDTO> getCasos() {
	return casos;
}

public void setCasos(List<CasoJoinDTO> casos) {
	this.casos = casos;
}

public List<PreguntaJoinDTO> getPreguntas() {
	return preguntas;
}

public void setPreguntas(List<PreguntaJoinDTO> preguntas) {
	this.preguntas = preguntas;
}
```

```
	public long getIdEx() {
		return idEx;
	}

	public void setIdEx(long idEx) {
		this.idEx = idEx;
	}

	/**
	 * @return the medico
	 */
	public MedicoJoinDTO getMedico() {
		return medico;
	}

	/**
	 * @param medico the medico to set
	 */
	public void setMedico(MedicoJoinDTO medico) {
		this.medico = medico;
	}

	/**
	 * @return the medico_save
	 */
	public MedicoSaveDTO getMedico_save() {
		return medico_save;
	}

	/**
	 * @param medico_save the medico_save to set
	 */
	public void setMedico_save(MedicoSaveDTO medico_save) {
		this.medico_save = medico_save;
	}
}
```

LoginAction.java

```
package mx.uam.azc.pt.actions;

import java.util.Map;

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

import org.apache.struts2.interceptor.SessionAware;

import com.opensymphony.xwork2.ActionSupport;

/**
 *
 * @author Rodrigo Gonzalez
 *
 */
public class LoginAction extends ActionSupport implements SessionAware {
```

```
	private UsuarioDTO usuario;
	/**
	 * Variable para el manejo de sesiones con struts
	 */
	private Map sessionData;
	private AdministradorUsuarios administradorUsuarios;

	/**
	 *  Metodo que regresa la cadena forma_login para el fin de la sesion en Struts
	 */
	public String forma_login() {
		sessionData.remove("usuario");
		return "forma_login";
	}

	/**
	 * Metodo que validad si un usuario se encuentra en el sistema y verifica que perfil posee dicho usuario
	 * @return Si el usuario se encontro y que perfil posee
	 * @throws Exception
	 */
	public String login() throws Exception {
		usuario = administradorUsuarios.leerUsuarioLoginPassword(usuario.getLogin(), usuario.getPassword());
		if (usuario == null) {
			addActionMessage("Usuario Incorrecto");
			sessionData.put("usuario", new UsuarioDTO());
			return "error";
		}
		sessionData.put("usuario", usuario);
		// NOOOOreturn "menu";
		if (usuario.getPerfil() == 1)
			return "menu";
		else if(usuario.getPerfil() == 3)
			return "menu_pacientes";
		else if(usuario.getPerfil() == 2)
			return "menu_medicos";
		else
			return "error";
	}

	/**
	 * Metodo que verifica si la sesion sigue activa
	 * @return Una redireccion al menu, un error o una redireccion a menu dependiendo del tipo de usuario
	 * @throws Exception
	 */
	public String menuRedirect() throws Exception {
		usuario = (UsuarioDTO) sessionData.get("usuario");
		if (usuario != null) {
			if (usuario.getPerfil() == 1 || usuario.getPerfil() == 2 || usuario.getPerfil() == 4 || usuario.getPerfil() == 3)
				return "menu";
			else
				return "forma_login";
		} else
```

```
                        return "error";
        }

        /**
         * @param sessionData the sessionData to set
         */
        public void setSessionData(Map sessionData) {
                this.sessionData = sessionData;
        }

        /**
         * @param sessionData the sessionData to set
         */
        public void setSession(Map sessionData) {
                this.sessionData = sessionData;
        }

        /**
         * @return the usuario
         */
        public UsuarioDTO getUsuario() {
                return usuario;
        }

        /**
         * @param usuario the usuario to set
         */
        public void setUsuario(UsuarioDTO usuario) {
                this.usuario = usuario;
        }

        /**
         * @return the sessionData
         */
        public Map getSessionData() {
                return sessionData;
        }

        /**
         * @return the administradorUsuarios
         */
        public AdministradorUsuarios getAdministradorUsuarios() {
                return administradorUsuarios;
        }

        /**
         * @param administradorUsuarios the administradorUsuarios to set
         */
        public void setAdministradorUsuarios(AdministradorUsuarios administradorUsuarios) {
                this.administradorUsuarios = administradorUsuarios;
        }
}
```

PrincipalAction.java

```
/**
 *
 */
```

```
package mx.uam.azc.pt.actions;

import com.opensymphony.xwork2.ActionSupport;

/**
 * @author Rodrigo Gonzalez
 * Clase utilizada para pruebas de funcionamiento básico del servidor (tomcat)
 */
public class PrincipalAction extends ActionSupport {

	@Override
	/**
	 * Metodo default para el uso de struts, siempre se ejecuta
	 */
	public String execute() throws Exception {

		return SUCCESS;
	}

}
```

AdministradorAdministradores.java

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

import java.util.List;

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

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, 01/01/2014
 * @author Rodrigo 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 );

}
```

AdministradorCasos.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, 01/01/2014
 * @author Rodrigo 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 );

}
```

AdministradorMedicos.java

```
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.FiltroMedicos;
import mx.uam.azc.pt.data.MedicoJoinDTO;

/**
 * Interfase que provee de los métodos para la administración de Médicos
 * @version 1.0, 01/01/2014
 * @author Rodrigo Gonzalez
 */
@Transactional( rollbackFor=Exception.class )
public interface AdministradorMedicos {

        /**
        * Inserta un médico al sistema
        * @param medico Datos del médico
        */
        public void insertarMedico (MedicoJoinDTO medico);

        /**
          * Actualiza un médico en el sistema
          * @param medico Datos del medico
          */
        public void actualizarMedico(MedicoJoinDTO medico);

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

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

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

AdministradorPacientes.java

```
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;

/**
 * Interfase que provee de los métodos para la administración de Pacientes
 * @version 1.0, 01/01/2014
 * @author Rodrigo 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 );

}
```

AdministradorPreguntas.java

```
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.FiltroCasos;
import mx.uam.azc.pt.business.local.FiltroPreguntas;
import mx.uam.azc.pt.data.CasoJoinDTO;
import mx.uam.azc.pt.data.PreguntaJoinDTO;
import mx.uam.azc.pt.data.PreguntaSaveDTO;
import mx.uam.azc.pt.data.RespuestaDTO;

/**
 * Interfase que provee de los métodos para la administración de Preguntas
 * @version 1.0, 01/01/2014
 * @author Rodrigo Gonzalez
 */
@Transactional( rollbackFor=Exception.class )
public interface AdministradorPreguntas {

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

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

        /**
          * Actualiza una pregunta en el sistema
          * @param pregunta Datos de la pregunta
          */
```

```
        public void actualizarPregunta(PreguntaJoinDTO pregunta);

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

        /**
          * 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 PreguntaJoinDTO 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 preguntas
          * @return La lista de DTOs con la información de las preguntas
          */
        @Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
        public List<PreguntaJoinDTO> 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 casos de acuerdo a un patron de busqueda
         * @param filter Filtro que fungira como patron de busqueda
         * @return La lista de preguntas que cumplieron con el patron de busqueda
         */
        @Transactional( propagation=Propagation.SUPPORTS, readOnly = true )
        public List<PreguntaJoinDTO> leerPreguntasFiltro( FiltroPreguntas filter );

}
```

AdministradorRespuestas.java

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

/**
 * Interfase que provee de los métodos para la administración de Respuestas
 * @version 1.0, 01/01/2014
 * @author Rorigo Gonzalez
 */
public interface AdministradorRespuestas {
```

```
}
```

AdministradorUsuarios.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, 01/01/2014
 * @author Rodrigo 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);

}
```

AdministradorAdministradores.java

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

import java.util.List;

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

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, 01/01/2014
 * @author Rodrigo 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 );

}
```

AdministradorCasos.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, 01/01/2014
 * @author Rodrigo 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 );

}
```

AdministradorMedicos.java

```
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.FiltroMedicos;
import mx.uam.azc.pt.data.MedicoJoinDTO;

/**
 * Interfase que provee de los métodos para la administración de Médicos
 * @version 1.0, 01/01/2014
 * @author Rodrigo Gonzalez
 */
@Transactional( rollbackFor=Exception.class )
public interface AdministradorMedicos {

	/**
	* Inserta un médico al sistema
	* @param medico Datos del médico
	*/
	public void insertarMedico (MedicoJoinDTO medico);

	/**
	  * Actualiza un médico en el sistema
	  * @param medico Datos del medico
	  */
	public void actualizarMedico(MedicoJoinDTO medico);

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

	/**
	  * Lee una lista de médicos
	  * @return La lista de DTOs con la información de los médicos
	  */
```

```
        @Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
        public List<MedicoJoinDTO> leerMedicos();

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

/**
 * Interfase que provee de los métodos para la administración de Pacientes
 * @version 1.0, 01/01/2014
 * @author Rodrigo 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 );

}
```

AdministradorPreguntas.java

```
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.FiltroCasos;
import mx.uam.azc.pt.business.local.FiltroPreguntas;
import mx.uam.azc.pt.data.CasoJoinDTO;
import mx.uam.azc.pt.data.PreguntaJoinDTO;
import mx.uam.azc.pt.data.PreguntaSaveDTO;
import mx.uam.azc.pt.data.RespuestaDTO;

/**
 * Interfase que provee de los métodos para la administración de Preguntas
 * @version 1.0, 01/01/2014
 * @author Rodrigo Gonzalez
 */
@Transactional( rollbackFor=Exception.class )
public interface AdministradorPreguntas {

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

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

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

        /**
          * Actualiza una respuesta en el sistema
          * @param respuesta Datos de la respuesta
```

```
     */
    public void actualizarRespuesta(RespuestaDTO respuesta);

    /**
     * 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 PreguntaJoinDTO 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 preguntas
     * @return La lista de DTOs con la información de las preguntas
     */
    @Transactional(propagation=Propagation.SUPPORTS, readOnly = true)
    public List<PreguntaJoinDTO> 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 casos de acuerdo a un patron de busqueda
     * @param filter Filtro que fungira como patron de busqueda
     * @return La lista de preguntas que cumplieron con el patron de busqueda
     */
    @Transactional( propagation=Propagation.SUPPORTS, readOnly = true )
    public List<PreguntaJoinDTO> leerPreguntasFiltro( FiltroPreguntas filter );

}
```

AdministradorRespuestas.java

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

/**
 * Interfase que provee de los métodos para la administración de Respuestas
 * @version 1.0, 01/01/2014
 * @author Rorigo Gonzalez
 */
public interface AdministradorRespuestas {

}
```

AdministradorUsuarios.java

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

```java
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, 01/01/2014
 * @author Rodrigo 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);

}
```

AdministradorAdministradoresImpl.java

```
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;

/**
 * Clase que provee la implementación de los métodos de la interfase
AdministradorAdministradores
 * @version 1.0, 01/01/2014
 * @author Rodrigo 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();
         }
}
```

AdministradorCasosImpl.java

```
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, 01/01/2014
 * @author Rodrigo 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();
	}
}
```

AdministradorMedicosImpl.java

```
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.AdministradorMedicos;
import mx.uam.azc.pt.data.MedicoJoinDTO;

/**
 * Clase que provee la implementación de los métodos de la interfase AdministradorMedicos
 * @version 1.0, 01/01/2014
 * @author Rodrigo Gonzalez
 */
public class AdministradorMedicosImpl implements
		AdministradorMedicos {

	/**
	 * @param medico
	 * Metodo que inserta a un médico a la Base de Datos
	 */
	public void insertarMedico(MedicoJoinDTO medico) {
		getSession().save(medico);
```

```
}

/**
  * Actualiza un médico en el sistema
  * @param medico Datos del medico
  */
public void actualizarMedico(MedicoJoinDTO medico) {

        getSession().update(medico);
}

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

        MedicoJoinDTO medico = new MedicoJoinDTO();
        medico = (MedicoJoinDTO) getSession().load(MedicoJoinDTO.class, id);
        return medico;
}

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

/**
 * Metodo que lee una lista de metodos de acuerdo a un filtro de busqueda
 */
public List<MedicoJoinDTO> leerMedicosFiltro(FiltroMedicos filter) {
        Session session = _sessionFactory.getCurrentSession();
        Query query = session.getNamedQuery("leer_medicos_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();
            }
}
```

AdministradorPacientesImpl.java

```
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, 01/01/2014
 * @author Rodrigo 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();
        }

}
```

AdministradorPreguntasImpl.java

```
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.AdministradorPreguntas;
```

```
import mx.uam.azc.pt.data.PreguntaJoinDTO;
import mx.uam.azc.pt.data.PreguntaSaveDTO;
import mx.uam.azc.pt.data.RespuestaDTO;

/**
 * Clase que provee la implementación de los métodos de la interfase
 * AdministradorPregunta
 *
 * @version 1.0, 01/01/2014
 * @author Rodrigo Gonzalez
 */
public class AdministradorPreguntasImpl implements AdministradorPreguntas {

	/**
	 * Inserta una pregunta en el sistema
	 *
	 * @param pregunta
	 *            Datos de la pregunta
	 */
	public void insertarPregunta(PreguntaJoinDTO 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 una pregunta en el sistema
	 *
	 * @param pregunta
	 *            Datos de la pregunta
	 */
	public void actualizarPregunta(PreguntaJoinDTO 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 una pregunta
 *
 * @param id
 *            Identificador de la pregunta
 * @return El DTO con los datos de la pregunta
 */
public PreguntaJoinDTO leerPregunta(long id) {

        PreguntaJoinDTO pregunta = new PreguntaJoinDTO();
        pregunta = (PreguntaJoinDTO) getSession().load(PreguntaJoinDTO.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 preguntas
 *
 * @return La lista de DTOs con la información de las preguntas
 */
public List<PreguntaJoinDTO> 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 preguntas de acuerdo a un filtro de busqueda
 */
```

```
	public List<PreguntaJoinDTO> leerPreguntasFiltro(FiltroPreguntas filter) {
		Session session = _sessionFactory.getCurrentSession();
		Query query = session.getNamedQuery("leer_preguntas_filtro");
		query.setString("pregunta", "%" + filter.getPregunta() + "%");
		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();
	}

}
```

AdministradorRespuestaImpl.java

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

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

import mx.uam.azc.pt.business.AdministradorRespuestas;

/**
 * Clase que provee la implementación de los métodos de la interfase AdministradorRespuestas
 * @version 1.0, 01/01/2014
 * @author Rodrigo Gonzalez
 */
public class AdministradorRespuestasImpl implements AdministradorRespuestas{



	/**
	  * 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();
        }


}
```

AdministradorUsuariosImpl.java

```
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, 01/01/2014
 * @author Rodrigo Gonzalez
 */
public class AdministradorUsuariosImpl implements AdministradorUsuarios {

        /**
          * 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();
        }
}
```

FiltroAdministradores.java

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

/**
 * Clase que actuara como filtro de busqueda para administradores
 * @author Rodrigo 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;
        }

}
```

FiltroCasos.java

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

/**
 * Clase que actuara como filtro de busqueda para casos
 * @author Rodrigo 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;
	}



}
```

FiltroMedicos.java

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

/**
 * Clase que actuara como filtro de busqueda medicos
 * @author Rodrigo Gonzalez
 *
 */
public class FiltroMedicos {

	/**
	 * Cadena que sera el patron de busqueda para el filtro de medicos
	 */
	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;
	}

}
```

FiltroPacientes.java

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

/**
 * Clase que actuara como filtro de busqueda para pacientes
```

```
 * @author Rodrigo 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;
	}

}
```

FiltroPreguntas.java

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

/**
 * Clase que actuara como filtro de busqueda para preguntas
 * @author Rodrigo Gonzalez
 *
 */
public class FiltroPreguntas {

	/**
	 * Cadena que sera el patron de busqueda para el filtro de pregutnas
	 */
	private String pregunta;

	/**
	 * @return the pregunta
	 */
	public String getPregunta() {
		return pregunta;
	}

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

```
}
```
AdministradorJoinDTO.java
```
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, 01/01/2014
 * @author Rodrigo Gonzalez
 */
@Entity
@Table(name = "administradores")
@PrimaryKeyJoinColumn(name="id")
@Proxy( lazy=false )
public class AdministradorJoinDTO extends UsuarioDTO implements Serializable {

	/**
	 * email alternativo del administrador, mapeado con el campo -email_aternativo- de la tabla -administradores- 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;
	}
}
```
AdministradorSaveDTO.java
```
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 Rodrigo Gonzalez
 */
@Entity
@Table(name = "administradores")
@PrimaryKeyJoinColumn(name="id")
@Proxy( lazy=false )
public class AdministradorSaveDTO extends UsuarioDTO implements Serializable {

	/**
	 * email alternativo del administrador, mapeado con el campo -email_aternativo- de la tabla
-administradores- 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;
	}
}
CasoJoinDTO.java
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, 01/01/2014
 * @author Rodrigo 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;
	}

}
```

CasoSaveDTO.java

```
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, 01/01/2014
 * @author Rodrigo 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;
	}

}
```

MedicoJoinDTO.java

```
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 en el sistema
 * @version 1.0, 01/01/2014
 * @author Rodrigo Gonzalez
 */
@Entity
@Table(name = "medicos")
@PrimaryKeyJoinColumn(name="id")
@Proxy( lazy=false )
public class MedicoJoinDTO extends UsuarioDTO implements Serializable{

	/**
	 * hospital del medico, mapeado con el campo -area- de la tabla -medicos- de la Base de Datos
	 */
	private String hospital;

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

	/**
	 * @return the puesto
	 */
	@Column(name="hospital")
	public String getHospital() {
		return hospital;
	}
	/**
	 * @param area the hospital to set
	 */
```

```
	public void setHospital(String hospital) {
		this.hospital = hospital;
	}
}
```

MedicoSaveDTO.java

```
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 en el sistema
 * @version 1.0, 01/01/2014
 * @author Rodrigo Gonzalez
 */
@Entity
@Table(name = "medicos")
@PrimaryKeyJoinColumn(name="id")
@Proxy( lazy=false )
public class MedicoJoinDTO extends UsuarioDTO implements Serializable{

	/**
	 * hospital del medico, mapeado con el campo -area- de la tabla -medicos- de la Base de
Datos
	 */
	private String hospital;

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

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

PacienteJoinDTO.java

```
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, 01/01/2014
 * @author Rodrigo 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;
	}
}
```

PacienteSaveDTO.java

```java
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, 01/01/2014
 * @author Rodrigo 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;
	}
}
```

PreguntaJoinDTO.java

```
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, 01/01/2014
 * @author Rodrigo Gonzalez
 */
@Entity @Table(name="preguntas")
@Proxy (lazy = false)
public class PreguntaJoinDTO 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;
//	}

}
```

PreguntaSaveDTO.java

```
package mx.uam.azc.pt.data;

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

```java
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, 01/01/2014
 * @author Rodrigo Gonzalez
 */
@Entity @Table(name="preguntas")
@Proxy (lazy = false)
public class PreguntaJoinDTO 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;
//      }

}
```

RespuestaDTO.java

```
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 una respuesta en el sistema, asociada a una pregunta de la clase PreguntaDTO
```

```
 * @version 1.0, 25/04/2012
 * @author Rodrigo 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;
	/**
	 * 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 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 status the status to set
         */
        public void setStatus(int status) {
                this.status = status;
        }

}
```

UsuarioDTO.java

```
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, 01/01/2014
 * @author Rodrigo 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;
    /**
     * 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;
    /**
     * email del usuario, mapeado con el campo -email- de la tabla -usuarios- de la Base de Datos
     */
    private String email;
    /**
     * 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;
    /**
     * 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 celular
	 */
	@Column(name = "celular")
	public String getCelular() {
		return celular;
	}
	/**
	 * @return the direccion
	 */
	@Column(name = "direccion")
	public String getDireccion() {
		return direccion;
	}
	/**
	 * @return the email
	 */
	@Column(name = "email")
	public String getEmail() {
		return email;
	}
	/**
	 * @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 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 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 email the email to set
 */
public void setEmail(String email) {
        this.email = email;
}
/**
 * @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 perfil the perfil to set
	 */
	public void setPerfil(int perfil) {
		this.perfil = perfil;
	}
	public void setStatus(int status) {
		this.status = status;
	}
}
```

AdministradoresInterceptor.java

```
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
 * @author Rodrigo Gonzalez
 *
 */
public class AdministradoresInterceptor implements Interceptor {

	public void destroy() {
		System.out.println("Destruyendo AdminsInterceptor...");
	}

	public void init() {
		System.out.println("Inicializando AdminsInterceptor...");
	}

	/**
	 * 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))
		if ((usuario.getPerfil() == 1))
			return invocation.invoke();
		else
			return "error";
	}
}
```

test.java

```
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.Map;
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.AdministradorMedicos;
import mx.uam.azc.pt.business.AdministradorPacientes;
import mx.uam.azc.pt.business.AdministradorPreguntas;
import mx.uam.azc.pt.business.AdministradorUsuarios;
import mx.uam.azc.pt.business.local.FiltroCasos;
import mx.uam.azc.pt.business.local.FiltroMedicos;
import mx.uam.azc.pt.business.local.FiltroPacientes;
import mx.uam.azc.pt.business.local.FiltroPreguntas;
import mx.uam.azc.pt.data.AdministradorJoinDTO;
import mx.uam.azc.pt.data.CasoJoinDTO;
import mx.uam.azc.pt.data.MedicoJoinDTO;
import mx.uam.azc.pt.data.PacienteJoinDTO;
import mx.uam.azc.pt.data.PreguntaJoinDTO;
import mx.uam.azc.pt.data.PreguntaSaveDTO;
import mx.uam.azc.pt.data.UsuarioDTO;

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

	/**
	 * @param args
	 * metodo main de java necesario para ejecutar una clase
	 */
	public static void main(String[] args) {
		try
	   {
	     new Test().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");
            //pruebaInsercionAdministrador();
            //pruebaLecturaAdministradoresTodos();
            //pruebaLecturayActualizacionAdministrador();
            //pruebaInsercionMedico();
            //pruebaLecturaMedicosTodos();
            //pruebaLecturayActualizacionMedicos();
            //pruebaLecturaMedicosFiltro();
            //pruebaInsercionPaciente();
            //pruebaLecturaPacientesTodos();
            //pruebaLecturayActualizacionPacientes();
            //pruebaLecturaPacientesFiltro();
            //pruebaInsercionCaso();
            //pruebaLecturaCasosTodos();
            //pruebaLecturaCasosFiltro();
            //pruebaLecturayActualizacionCaso();
            //pruebaInsercionPregunta();
            pruebaLecturaPreguntasFiltro();
            //pruebaLecturayActualizacionPregunta();
          System.out.println("acabo");
        }

    private void pruebaUsuarios(){
            AdministradorUsuarios administrador = getAdministradorUsuarios();
            //Map<Long,String> mapa = administrador.cargarUsuarios();
            //System.out.println(mapa.size());
            UsuarioDTO usuario = new UsuarioDTO();
            usuario = administrador.leerUsuarioLoginPassword("sandy", "sandy");
    }

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

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

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

            AdministradorAdministradores administrador = getAdministradorAdministradores();
            AdministradorJoinDTO admin = new AdministradorJoinDTO();
            admin = administrador.leerAdministrador(2);
            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 caso de la base de datos
	 */
	private void pruebaLecturayActualizacionCaso(){

		AdministradorCasos administrador = getAdministradorCasos();
		CasoJoinDTO caso = new CasoJoinDTO();
		caso = administrador.leerCaso(1);
		System.out.println("ID: " + caso.getId() + " | Titulo: "  + caso.getTitulo());
		caso.setTitulo("el que sea");
		System.out.println("Nuevo Titulo: " + caso.getTitulo());
		administrador.actualizarCaso(caso);
	}

	/**
	 * metodo que prueba la lectura y actualizacion de una pregunta de la base de datos
	 */
	private void pruebaLecturayActualizacionPregunta(){

		AdministradorPreguntas administrador = getAdministradorPreguntas();
		PreguntaJoinDTO pregunta = new PreguntaJoinDTO();
		pregunta = administrador.leerPregunta(1);
		System.out.println("ID: " + pregunta.getId_preguntas() + " | Pregunta: "  +
pregunta.getPregunta());
		pregunta.setPregunta("pregunta nueva");
		System.out.println("Nueva Pregunta: " + pregunta.getPregunta());
		administrador.actualizarPregunta(pregunta);
	}

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

		AdministradorAdministradores administrador = getAdministradorAdministradores();

		AdministradorJoinDTO ad = new AdministradorJoinDTO();

		ad.setAp_materno("Gonzalez");
		ad.setAp_paterno("Hernandez");
		ad.setCelular("23523523235");
		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.setLogin("admin");
		ad.setNombres("Rodrigo");
		ad.setPassword("admin");
		ad.setTel_casa("55779988");
		ad.setPerfil(1);

		administrador.insertarAdministrador(ad);
```

```
}

private void pruebaInsercionPregunta(){
        AdministradorPreguntas administrador = getAdministradorPreguntas();
        PreguntaJoinDTO pregunta = new PreguntaJoinDTO();
        pregunta.setPregunta("¿Por que asdasdasda?");
        administrador.insertarPregunta(pregunta);
}

private void pruebaInsercionCaso(){
        AdministradorCasos administrador = getAdministradorCasos();
        CasoJoinDTO caso = new CasoJoinDTO();
        caso.setContenido_caso("contenido del caso");
        caso.setTitulo("Caso 1");
        administrador.insertarCaso(caso);
}

private void pruebaInsercionMedico(){
        AdministradorMedicos medico = getAdministradorMedicos();

        MedicoJoinDTO med = new MedicoJoinDTO();

        med.setAp_materno("asdasdas");
        med.setAp_paterno("efsdfsdf");
        med.setCelular("22243333");
        med.setDireccion("direcionmedico");
        med.setEmail("medico@medicos.com");
        Date fecha = new Date();
        med.setFecha_ingreso(fecha);
        med.setFecha_nacimiento(fecha);
        med.setLogin("medico1");
        med.setNombres("Juancho");
        med.setPassword("qazwsx");
        med.setTel_casa("55566677");
        med.setPerfil(2);
        med.setHospital("Hospital 1");
        medico.insertarMedico(med);
}

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

        AdministradorMedicos medico = getAdministradorMedicos();
        List<MedicoJoinDTO> lista = new ArrayList<MedicoJoinDTO>();
        lista = medico.leerMedicos();
        System.out.println("Tamaño: " + lista.size());
}

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

        AdministradorCasos caso = getAdministradorCasos();
```

```
        List<CasoJoinDTO> lista = new ArrayList<CasoJoinDTO>();
        lista = caso.leerCasos();
        System.out.println("Tamaño: " + lista.size());
}

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

        AdministradorMedicos administrador = getAdministradorMedicos();
        MedicoJoinDTO med = new MedicoJoinDTO();
        med = administrador.leerMedico(4);
        System.out.println("ID: " + med.getId() + " | Hospital: "  + med.getHospital());
        med.setHospital("Hospital222233");
        System.out.println("HOSPITAL NUEVO: " + med.getHospital());
        administrador.actualizarMedico(med);
}

private void pruebaLecturaMedicosFiltro(){
        AdministradorMedicos administrador = getAdministradorMedicos();
        FiltroMedicos filter = new FiltroMedicos();
        filter.setNombres("juan");
        List<MedicoJoinDTO> lista_medicos = new ArrayList<MedicoJoinDTO>();
        lista_medicos = administrador.leerMedicosFiltro(filter);
        System.out.println(lista_medicos.size());
}

private void pruebaInsercionPaciente(){
        AdministradorPacientes paciente = getAdministradorPacientes();

        PacienteJoinDTO pa = new PacienteJoinDTO();

        pa.setAp_materno("asdasdas");
        pa.setAp_paterno("efsdfsdf");
        pa.setCelular("22243333");
        pa.setDireccion("direcionmedico");
        pa.setEmail("medico@medicos.com");
        Date fecha = new Date();
        pa.setFecha_ingreso(fecha);
        pa.setFecha_nacimiento(fecha);
        pa.setLogin("medico1");
        pa.setNombres("Juancho");
        pa.setPassword("qazwsx");
        pa.setTel_casa("55566677");
        pa.setPerfil(3);
        pa.setEnfermedades("sida");
        pa.setNo_afiliacion("000001");
        pa.setTipo_sangre("blah");
        pa.setAlergias("alerer");

        paciente.insertarPaciente(pa);
}

/**
 * metodo que prueba la lectura de todos los pacientes de la base de datos
```

```
 */
private void pruebaLecturaPacientesTodos(){

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

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

        AdministradorPacientes administrador = getAdministradorPacientes();
        PacienteJoinDTO pa = new PacienteJoinDTO();
        pa = administrador.leerPaciente(8);
        System.out.println("ID: " + pa.getId() + " | Enfermedad: "  + pa.getEnfermedades());
        pa.setEnfermedades("gonorrea");
        System.out.println("neuva enermedad: " + pa.getEnfermedades());
        administrador.actualizacionPaciente(pa);
}

private void pruebaLecturaCasosFiltro(){
        AdministradorCasos administrador = getAdministradorCasos();
        FiltroCasos filter = new FiltroCasos();
        filter.setTitulo("aso");
        List<CasoJoinDTO> lista_casos = new ArrayList<CasoJoinDTO>();
        lista_casos = administrador.leerCasosFiltro(filter);
        System.out.println(lista_casos.size());
}

private void pruebaLecturaPreguntasFiltro(){
        AdministradorPreguntas administrador = getAdministradorPreguntas();
        FiltroPreguntas filter = new FiltroPreguntas();
        filter.setPregunta("or");
        List<PreguntaJoinDTO> lista_preguntas = new ArrayList<PreguntaJoinDTO>();
        lista_preguntas = administrador.leerPreguntasFiltro(filter);
        System.out.println(lista_preguntas.size());
}

private void pruebaLecturaPacientesFiltro(){
        AdministradorPacientes administrador = getAdministradorPacientes();
        FiltroPacientes filter = new FiltroPacientes();
        filter.setNombres("juan");
        List<PacienteJoinDTO> lista_pacientes = new ArrayList<PacienteJoinDTO>();
        lista_pacientes = administrador.leerPacientesFiltro(filter);
        System.out.println(lista_pacientes.size());
}

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

   return (AdministradorAdministradores)context.getBean( "administradorAdministradores"
```

```
);
        }

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

       return (AdministradorMedicos)context.getBean( "administradorMedicos" );
     }

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

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

    private AdministradorUsuarios getAdministradorUsuarios()
    {
            ApplicationContext context = new ClassPathXmlApplicationContext
            ("spring-services.xml");
            return (AdministradorUsuarios)context.getBean("administradorUsuarios");
    }

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

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

}
```

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

        </package>
</struts>
```

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_administradores_filtro">
                FROM AdministradorJoinDTO ad WHERE (ad.nombres LIKE :nombres) AND
ad.status=0
        </query>
        <query name="leer_administradores">
                FROM AdministradorJoinDTO ad
        </query>
        <query name="leer_medicos_filtro">
                FROM MedicoJoinDTO me WHERE (me.nombres LIKE :nombres) AND
me.status=0
        </query>
        <query name="leer_medicos">
```

```
                FROM MedicoJoinDTO me
        </query>
        <query name="leer_pacientes_filtro">
                FROM PacienteJoinDTO pa WHERE (pa.nombres LIKE :nombres) AND
pa.status=0
        </query>
        <query name="leer_pacientes">
                FROM PacienteJoinDTO pa
        </query>
        <query name="leer_casos_filtro">
                FROM CasoJoinDTO ca WHERE (ca.titulo LIKE :titulo) AND ca.status=0
        </query>
        <query name="leer_casos">
                FROM CasoJoinDTO ca
        </query>
        <query name="leer_preguntas_filtro">
                FROM PreguntaJoinDTO pr WHERE (pr.pregunta LIKE :pregunta) AND pr.status=0
        </query>
        <query name="leer_preguntas">
                FROM PreguntaJoinDTO pr
        </query>
        <query name="leer_usuario_login_pass">
                FROM UsuarioDTO u WHERE
                (
                (u.login = ? )AND
                (u.password = ?)
                )
        </query>

</hibernate-mapping>
```

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/ptForo?autoReconnect=true</value>
</property>
<property name="maxWait" value="10000"/>
<property name="username" value="root"/>
```

```
<property name="password" value="amenaza"/>
</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.AdministradorJoinDTO</value>
<value>mx.uam.azc.pt.data.AdministradorSaveDTO</value>
<value>mx.uam.azc.pt.data.MedicoJoinDTO</value>
<value>mx.uam.azc.pt.data.MedicoSaveDTO</value>
<value>mx.uam.azc.pt.data.PacienteSaveDTO</value>
<value>mx.uam.azc.pt.data.PacienteJoinDTO</value>
<value>mx.uam.azc.pt.data.CasoSaveDTO</value>
<value>mx.uam.azc.pt.data.CasoJoinDTO</value>
<value>mx.uam.azc.pt.data.PreguntaSaveDTO</value>
<value>mx.uam.azc.pt.data.PreguntaJoinDTO</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>
Spring-services.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"
>
<import resource="spring-dev.xml"/>
<bean id="administradorAdministradores"
class="mx.uam.azc.pt.business.local.AdministradorAdministradoresImpl"></bean>
```

```
<bean id="administradorUsuarios"
class="mx.uam.azc.pt.business.local.AdministradorUsuariosImpl"></bean>
<bean id="administradorMedicos"
class="mx.uam.azc.pt.business.local.AdministradorMedicosImpl"></bean>
<bean id="administradorPacientes"
class="mx.uam.azc.pt.business.local.AdministradorPacientesImpl"></bean>
<bean id="administradorCasos"
class="mx.uam.azc.pt.business.local.AdministradorCasosImpl"></bean>
<bean id="administradorPreguntas"
class="mx.uam.azc.pt.business.local.AdministradorPreguntasImpl"></bean>
</beans>
```

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">/WEB-
INF/pages/administradores/listar_medicos.jsp
                        </result>
                        <result name="ver_medicos">/WEB-
INF/pages/administradores/ver_medico.jsp
                        </result>
                        <result name="forma_insertar_medicos">/WEB-
```

```
INF/pages/administradores/forma_insertar_medicos.jsp
                        </result>
                        <result name="forma_actualizar_medicos">/WEB-
INF/pages/administradores/forma_actualizar_medicos.jsp
                        </result>
                        <result name="listar_medicos_redirect"
type="redirectAction">administradores_listar_medicos
                        </result>
                        <result name="ver_medicos_redirect" type="redirectAction">medicos_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_preguntas">/WEB-
INF/pages/administradores/listar_preguntas.jsp
                        </result>
                        <result name="ver_preguntas">/WEB-
INF/pages/administradores/ver_pregunta.jsp
                        </result>
                        <result name="forma_insertar_preguntas">/WEB-
INF/pages/administradores/forma_insertar_preguntas.jsp
                        </result>
                        <result name="forma_actualizar_preguntas">/WEB-
INF/pages/administradores/forma_actualizar_preguntas.jsp
                        </result>
                        <result name="listar_preguntas_redirect"
type="redirectAction">administradores_listar_preguntas
                        </result>
                        <result name="ver_preguntas_redirect"
type="redirectAction">preguntas_ver
                        </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">/WEB-
INF/pages/medicos/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>
```

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;
      }
}
```

style.css

```
body {
      background-color: #F8FAB3;
      font-family: sans-serif,Helvetica;
}

: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: #91BAB6;
}

.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,textarea.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;
}
```

principal.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>

HOLA MUNDO CRUEL web-inf!!

<a href="administradores_menu.action">Menu Administrador</a>
</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>Aplicación Foro</display-name>
```

```
  <filter>
<filter-name>struts-cleanup</filter-name>
<filter-class>
org.apache.struts2.dispatcher.ActionContextCleanUp
</filter-class>
</filter>
<filter-mapping>
<filter-name>struts-cleanup</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>

  <filter>
    <display-name>SiteMeshFilter</display-name>
    <filter-name>SiteMeshFilter</filter-name>
    <filter-class>com.opensymphony.module.sitemesh.filter.PageFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>SiteMeshFilter</filter-name>
    <url-pattern>*.action</url-pattern>
  </filter-mapping>

  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>principal.jsp</welcome-file>
  </welcome-file-list>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring-services.xml</param-value>
  </context-param>
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <listener>
    <listener-class>mx.uam.azc.pt.InicializadorCatalogos</listener-class>
  </listener>

</web-app>
```

Conclusiones

El uso de este prototipo de Sistema de apoyo para pacientes de enfermedades Neoplásicas será muy útil si se implementa en un buen número de instituciones médicas ya que éstas carecen de plataforas de este tipo, en las que se puede gestionar de manera más adecuada a los usuarios, sin importar que los médicos y pacientes se encuentren alejados físicamente. Asi como también los pacientes

podrán compartir sus experiencias con otros médicos o pacientes que tengan los mismos padecimientos que ellos.

El asesor de este proyecto intervino siempre de forma adecuada siempre que se requirió su apoyo resolviendo dudas, recomendando referencias y libros. Tambien proporciono todo su conocimiento e interés en este Proyecto Terminal.

Documentación de referencia.

Bibliografía

[I] “E Learning Knowledge”
http://elearning.azc.uam.mx/moodle
Fecha de Consulta: 18-Marzo-2009

[II] “Breast Cancer Network of Strength (Formerly Y-ME National Breast Cancer Organization)”
http://www.networkofstrength.org/espanol/
Fecha de Consulta: 07-Febrero-2009

[III] “Enciclopedia Médica”
http://www.ferato.com/wiki/index.php/Neoplasia
Fecha de Consulta: 01-Febrero-2009

[IIIV] “Aplicación Web”
http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web
Fecha de consulta: 25- Febrero-2009

[V] “Introducción al Framework de Struts”
http://www.scribd.com/doc/97147/introduccion-al-framework-struts
Fecha de Consulta: 27-Febrero-2009

[VI] “Tutorial JSP’s, Java Servers Pages”
http://geneura.ugr.es/~jmerelo/JSP/
Fecha de Consulta: 27-Febrero-2009

[VII] “**Administración y Gestión de un Servidor Web Apache**”
http://akira.azul.googlepages.com/apache.pdf
Fecha de Consulta: 28-Febrero-2009

[VIII] “Tomcat”
http://es.wikipedia.org/wiki/Apache_Tomcat
Fecha de Consulta:1-Marzo-2009