SISTEMA DE CONTROL DE CALIDAD EN EQUIPOS DE MASTOGRAFÍA

Agustín Fuentevilla Perera
Ingeniería en Computación
No. Cuenta. 9351704-7

Elizabeth Morales Morato
Ingeniería en Computación
No. Cuenta. 9534476-2

Director de Tesis
Ing. Sergio Noble Camargo

INDICE

Gracias a mi esposa

Por la paciencia y todos los años de felicidad que me has brindado, además gracias por la hermosa familia que hemos formado juntos. Te agradezco que nunca te hayas soltado de la cuerda y siempre hayas sido ese ser especial que me ha apoyado para crecer. Y recuerda que para atrás nunca, ni siquiera para tomar impulso.

Gracias a mis padres

Porque ustedes me dieron el ejemplo para estudiar una carrera y convertirme en un universitario productivo. Gracias por los valores que me inculcaron. Y en especial, gracias Mamá por haberme apoyado en toda mi vida escolar y SIEMPRE haber estado ahí.

Gracias a mi hermana

Que en mi infancia fue mi compañera y mi cómplice. Gracias por haberme aguantado y por apoyarme en estos últimos años y gracias por haberme apoyado con esas imágenes.

Y principalmente, gracias a mi hija

Cuya existencia me hace la persona más feliz del planeta. Gracias por habernos escogido hija mía. Te amo y recuerda que siempre eres mi motor para buscar cosas mejores.

Gracias Sergio

Porque después de todos estos años, no solo fuiste mi Jefe y mi sinodal, también has sido un gran amigo. Que el futuro nos depare abundancia y éxito amigo mío.

Gracias Dios

Por las bendiciones que me has otorgado. Eres mi faro en esos días nubosos. Gracias Cristo Señor, gracias por haberte mantenido tan cerca aunque en algunas ocasiones yo estuve tan lejos, y te agradezco la familia que me has otorgado.

Agustín Fuentevilla Perera

Gracias a Dios

Por la vida, por guiarme e iluminar mí camino, por todos los seres de luz que me has enviado para ayudarme a crecer y tener paz, al haberme dado las fuerzas para levantarme, enseñándome que la vida está llena de oportunidades para disfrutarse en cada momento. Gracias por ayudarme a comprender que el éxito viene acompañado del fracaso, y que esté nos fortalece para llegar más alto cada vez con él aprendizaje otorgado. Por permitirme soñar y esperar lo mejor de la humanidad.

Gracias a mi Padre

Porqué con su sabiduría, amor, paciencia y cariño ha sabido guiarme para lograr mis sueños.

Gracias a cada uno de mis hermanos

Por ser mis amigos, compañeros de juego, cuidarme, amarme, respetarme y acompañarme en cada una de las etapas de mi vida, quiero decirles que me inspiran a ser mejor cada día.

Gracias a mi familia

Porque me ayudo a formar raíces fuertes como su humildad y fortaleza que me acompañan.

Gracias a mi amigos

Por su cariño, comprensión, amor, coraje porque fortalecen mi espíritu para seguir adelante, gracias por formar y ser parte de mi vida y por dejar de una parte de su ser y luz en mí.

Gracias a la Universidad Nacional Autónoma de México

Por la oportunidad de creer en mí para lograr mis sueños y el espacio y tiempo compartido.

Gracias Sergio

Por ser un amigo y por ser un vínculo para llegar a este sueño cumplido.

Gracias Agustín

Por ser mi compañero de tesis para cumplir este objetivo.

…… *GRACIAS*

Elizabeth Morales Morato

Resumen

Se describe la solución de un “Sistema de Control de Calidad en Equipos de Mastografía”, para el “Centro de Equidad de Género y Salud Reproductiva” de la “Secretaria de Salud” en México. En esta solución se plantea la gestión de diferentes protocolos o pruebas diseñadas en el “Manual de Control de Calidad en Mastografía”, que definen como evaluar equipos de mastografía, material e instalaciones entre otros, con la finalidad de monitorear el funcionamiento de los mismos y con ello mantener nivel optimo de los equipos y/o materiales para garantizar la calidad de la imagen. Se describe el desarrollo de un sistema web para facilitar la recolección de información, divido en diferentes módulos como: catálogos, administración de usuarios, administración de las pruebas, generación física de las pruebas, control y monitoreo de pruebas.

Abstract

This description is about the solution of a “Sistema de Control de Calidad en Equipos de Mastografía”, for “Centro de Equidad de Género y Salud Reproductiva”, of “Secretaria de Salud” in México. In this solution poses different management protocols or tests designed in the “Manual de Control de Calidad en Mastografía”, which define how to evaluate mammography equipment, materials and facilities among others, in order to monitor the performance of the same and thus maintain optimum level of equipment and/or materials to ensure the quality of the image. This description is the development of a web system to facilitate the collection of information, divided into different modules such as: catalogs, user management, test management, test physical generation, control and monitoring of evidence.

SISTEMA DE CONTROL DE CALIDAD EN EQUIPOS DE MASTOGRAFÍA

Objetivo

Crear el Sistema de Control de Calidad en Equipos de Mastografía (SCCEM), que apoyará al "Programa de Garantía de Calidad" en el control de los equipos, estableciendo las pruebas e indicadores para evaluar cada uno de ellos. Las pruebas se realizan de forma periódica a los equipos, los cuales tendrán criterios de aceptación que permiten calificar los resultados de cada prueba.

Esto mantendrá el nivel óptimo de los equipos y ayudará a identificar las fallas y acciones que se deben tomar en caso de identificar errores, así como tomar decisiones de manera oportuna que logren mejorar el servicio de salud que es necesario para la detección temprana o el diagnóstico oportuno al momento de hacerse una mastografía.

Los responsables de la correcta operación de este sistema serán el personal Médico Especializado en Radiología, los Técnicos Radiólogos y Físicos Médicos.

Capítulo I - Introducción

Considerando que nuestra sociedad es sumamente heterogénea, cambiante, creciente, multifacética, complicada; por sus poblaciones, costumbres, religión, o por su etnia, también por sus muy diversas condiciones geográficas o climáticas, nos lleva a pensar qué la nuestra son muchas sociedades en una.

En México, desde hace décadas, se ha observado un cambio en la manera de enfermarse y de morir, hoy predominan las enfermedades no transmisibles y las lesiones, esta transición se asocia al envejecimiento de la población y al creciente desarrollo de riesgos relacionados con los estilos de vida poco saludables. La transformación de los patrones de daños a la salud impone retos en la manera de organizar y gestionar los servicios, porque las etapas intermedias y terminales de las enfermedades que nos aquejan demandan una atención compleja, otras son de larga duración y costosas, que exigen el empleo de alta tecnología y la participación precisa de múltiples áreas de especialidad.

Ante tales circunstancias, la protección de la salud de los mexicanos requiere de estrategias integrales, que fortalezcan y amplíen la lucha contra los riesgos sanitarios, además de favorecer una cultura de la salud y así como el desarrollo de oportunidades para elegir estilos de vida saludables. La política nacional debe incidir sobre los riesgos críticos de la salud y adaptar los servicios de todo el sector a las nuevas necesidades, promoviendo altos niveles de calidad, seguridad y eficiencia.

Después de un amplio diagnóstico de la situación nacional, se han identificado los ejes rectores que guiarán los esfuerzos del Sector Salud para dar cabal respuesta a las demandas más vivas y legítimas de la población como:

1. Dar énfasis a la promoción de la salud y la prevención de enfermedades.
2. Garantizar el aseguramiento universal, con el propósito de que cada mexicano tenga acceso a servicios integrales de salud.
3. Garantizar que los bienes y servicios estén libres de riegos sanitarios.
4. Suministrar oportunamente los medicamentos e insumos requeridos.
5. Brindar una atención de calidad, con calidez y asegurar a toda la población.
6. Fortalecer la infraestructura y el equipamiento médico para ofrecer a los pacientes una atención efectiva en sus lugares de origen.

La Salud es sin duda una prioridad para el desarrollo de la sociedad, es como alcanzar una adecuada calidad de vida. La salud no sólo entendida como la atención al enfermo, sino como un “ciclo perfecto” que brinde atención clínica (médica, quirúrgica, terapéutica, etc.); capaz de brindar información coherente y concisa garantizando un correcto desenvolvimiento y reincorporación a la

vida social; un enorme cúmulo de recursos físicos, materiales, técnicos y humanos dedicados a la salud; que sea una estructura fiable de cobertura con alcances nacionales.

Tomando en cuenta estas realidades y consideraciones, el Plan Nacional de Desarrollo 2007-2012 propone, en materia de salud, avanzar hacia la universalidad en el acceso a servicios médicos de calidad a través de una integración funcional y programática de las instituciones públicas bajo la rectoría de la Secretaría de Salud. Sé diseñó el Programa Nacional de Salud 2007-2012, para cumplir este compromiso, el cual está estructurado en cinco grandes objetivos de política social:

1. Mejorar las condiciones de salud de la población.
2. Brindar servicios de salud eficiente, con calidad, calidez y seguridad para el paciente.
3. Reducir las desigualdades en salud mediante intervenciones focalizadas en comunidades marginadas y grupos vulnerables.
4. Evitar el empobrecimiento de la población por motivos de salud mediante el aseguramiento médico universal.
5. Garantizar que la salud contribuya a la superación de la pobreza y al desarrollo humano en México.

Nos enfocaremos en los siguientes objetivos por su relevancia en el ámbito de la calidad en los servicios de salud:

Meta 1

Como mejorar las condiciones de salud de la población, por medio de reformas en los valores de indicadores de salud asociados a padecimientos o grupos de edad específicos, como la mortalidad por enfermedades, y/o indicadores de salud como los eventuales. Para este objetivo en particular se seleccionaron siete metas estratégicas en indicadores de ambos tipos, que son las siguientes:

i) **Meta 1.1** Aumentar la esperanza de vida al nacer 1.5 años.
ii) **Meta 1.2** Disminuir 15% la mortalidad por enfermedades del corazón en la población menor de 65 años.
iii) **Meta 1.3** Reducir 20% la velocidad de crecimiento de la mortalidad por diabetes mellitus con respecto a la tendencia observada entre 1995-2006.
iv) **Meta 1.4** Disminuir en 10% la prevalencia de consumo, por primera vez, de drogas ilegales en la población de 12 a 17 años de edad.
v) **Meta 1.5 Incrementar al triple la cobertura en el último año de detección de cáncer de mama por mastografía en mujeres de 50 a 69 años**.
vi) **Meta 1.6** Disminuir 27% la tasa de mortalidad por cáncer cérvico-uterino por 100,000 mujeres de 25 años.
vii) **Meta 1.7** Reducir 15% el número de muertes causadas por accidentes de tránsito de vehículos de motor en población de 15 a 29 años de edad.

Meta 2

Prestar servicios de salud con calidad y seguridad, para mejorar las condiciones de salud de una población es indispensable contar con servicios personales y de salud pública de calidad y seguros, que respondan a las expectativas de los usuarios y tomen en consideración su diversidad cultural.

i) **Meta 2.1** Acreditar el 100% de las unidades de salud que ofrecen servicios al Sistema de Protección Social en Salud (SPSS).

Para el cumplimento de estos objetivos y metas, se diseñaron las siguientes estrategias asociadas a las funciones sustantivas del Sistema Nacional de Salud: la rectoría efectiva, como la prestación de servicios con calidad y seguridad, financiamiento equitativo y sostenible, con la generación de recursos suficientes y oportunos.

Los responsables como instrumentos de la ejecución de las líneas de acción y actividades comprometidas en el Programa Sectorial de Salud 2007-2012, se encuentran contenidas en el Reglamento Interior vigente de la Secretaria de Salud y los correspondientes a las instituciones y dependencias que conforman el Sector Salud.

Las estrategias son las siguientes:

1. Fortalecer y modernizar la protección contra riesgos sanitarios;
2. Fortalecer e integrar las acciones de promoción de la salud, y prevención y control de enfermedades;
3. **Situar la calidad en la agenda permanente del Sistema Nacional de Salud**;
4. Desarrollar instrumentos de planeación, gestión y evaluación para el Sistema Nacional de Salud;
5. Organizar e integrar la prestación de servicios del Sistema Nacional de Salud;
6. Garantizar recursos financieros suficientes para llevar a cabo las acciones de protección contra riesgos sanitarios y promoción de la salud;
7. Consolidar la reforma financiera para hacer efectivo el acceso universal a los servicios de salud a la persona;
8. Promover la inversión en sistemas, tecnologías de la información y comunicaciones que mejoren la eficiencia y la integración del sector;
9. Fortalecer la investigación y la enseñanza en salud para el desarrollo del conocimiento y los recursos humanos, y
10. Apoyar la prestación de servicios de salud mediante el desarrollo de la infraestructura y el equipamiento necesarios.

Líneas de acción de la estrategia 3:

i) Implantar el Sistema Integral de Calidad en Salud (SICALIDAD) en el Sistema Nacional de Salud,
ii) Incorporar programas de calidad en la formación académica de técnicos y profesionales de la salud,
iii) Impulsar la utilización de las guías de práctica clínica y protocolos de atención médica,
iv) Promover políticas interculturales de respeto a la dignidad y derechos humanos de las personas,
v) Proteger los derechos de los pacientes mediante el arbitraje y la conciliación,
vi) Diseñar e instrumentar una Política Nacional de Medicamentos que promueva el desarrollo de modelos para el suministro eficiente y oportuno de medicamentos e insumos para la salud,
vii) Fortalecer la vinculación de la bioética con la atención médica, y
viii) Actualizar el marco jurídico en materia de servicios de atención médica.

Líneas de acción de la estrategia 9:

i) Definir la agenda de investigación y desarrollo con base en criterios de priorización en salud.
ii) Reorientar la innovación tecnológica y la investigación para la salud hacia los padecimientos emergentes, las enfermedades no transmisibles y las lesiones.
iii) Impulsar la formación de recursos humanos especializados de acuerdo a las proyecciones demográficas y epidemiológicas.
iv) Incentivar el desarrollo y distribución nacional del capital humano especializado con base en las necesidades regionales de atención a la salud.
v) Desarrollar competencias gerenciales en el personal directivo que fortalezcan la toma de decisiones en salud.

Líneas de acción de la estrategia 10:

i) Asegurar la producción nacional de reactivos, vacunas y otros dispositivos médicos estratégicos para la seguridad nacional.
ii) Impulsar la infraestructura de prevención y control de enfermedades, a través del fortalecimiento de la capacidad estructural de inteligencia para emergencias en salud, la reingeniería de los laboratorios de diagnóstico y referencia y la creación de una planta de producción de vacunas de BIRMEX.
iii) Impulsar la dignificación y el mantenimiento de la infraestructura y equipo industrial de las unidades de salud.
iv) Impulsar el financiamiento y establecimiento de políticas para la renovación y mantenimiento de equipo médico mediante el desarrollo de áreas de ingeniería biomédica en las unidades de atención a la salud.
v) Promover la creación de centros de atención especializada para pacientes ambulatorios (UMAES y UNEMES) y nuevas unidades de atención hospitalaria, con modelos innovadores de financiamiento.

Estas líneas de acción nos llevan a la implementación de diversos programas para mejorar la calidad de los Servicios de Salud en México.

En la evolución de la historia de calidad de salud en la atención en México, se mencionan los siguientes:

1.	**AUDITORÍAS MÉDICAS IMSS (1950´s).**
2.	**CÍRCULOS DE CALIDAD INPER (1985).**
3.	**EVALUACIÓN DE LA CALIDAD DE LA ATENCIÓN MÉDICA.**
4.	**FINANCIAMIENTO DE PROYECTOS DE MEJORA CONTINUA PASSPA (1994).**
5.	**PROGRAMA INTEGRADO DE CALIDAD IMSS (1997).**
6.	**PROGRAMA DE MEJORA CONTINUA DE LA CALIDAD EN LA ATENCIÓN MÉDICA IMSS (1987) SSA (1997-2000).**
7.	**CRUZADA NACIONAL POR LA CALIDAD DE LOS SERVICIOS DE SALUD (2001-2006).**
8.	**SISTEMA INTEGRAL DE CALIDAD EN SALUD (SICALIDAD 2007-2012).**

Fuente: Dirección General de Calidad y Educación en Salud, SSA. [1]

Un programa pionero fue el de la Cruzada Nacional por la Calidad de los Servicios de Salud, puesta en marcha hacia finales del 2001, que permitió el rápido posicionamiento en el Sector Salud, de dos elementos fundamentales: trato digno y atención médica efectiva, eficiente, ética y segura. En su implantación enfrentó obstáculos tales como la resistencia al cambio, deficiencia e insuficiencia de personal y limitaciones en la operación de sus líneas de acción.

Para cambiar los paradigmas de los Servicios de Salud se buscó situar la calidad en la agenda permanente del Sistema Nacional de Salud aplicando sus respectivas líneas de acción.

El Programa de Acción Especifico del Sistema Integral de Calidad en Salud (SICALIDAD), diagnostica nuestras carencias e insuficiencias en relación a la calidad de los servicios de salud; formulando propuestas y acciones tanto para mantener los avances logrados, como para favorecer un programa de mejora continua que sitúe la calidad en la agenda permanente en las organizaciones de salud.

El compromiso para la mejora continua es impulsar diversas acciones y programas desde la perspectiva de la calidad percibida, la calidad técnica–seguridad del paciente y la institucionalización de la calidad. A través de acciones, proyectos, lineamientos e instrumentos enfocados en los usuarios, los profesionales de la salud y las organizaciones, contribuyen a elevar e innovar la calidad de los servicios y la seguridad del paciente en el Sistema Nacional de Salud.

El objetivo central es generar confianza en los ciudadanos en las organizaciones de salud e impulsar la mejora en la calidad de la atención de la salud, garantizando la seguridad de los pacientes.

[1] Histórico de los programas más relevantes para implementar la calidad en los servicios de salud en México.

Con base en la cooperación de los profesionales de la salud, es necesario buscar organizaciones de excelencia, sensibles a las necesidades y expectativas de los ciudadanos, que hagan de la calidad y seguridad del paciente su agenda y preocupación cotidiana.
Con esta finalidad, el Programa de Acción Específico 2007-2012 SICALIDAD incluye propuestas innovadoras y la integración de esfuerzos que permiten posicionar a la calidad de los servicios de salud como un tema permanente en la gestión de las unidades de salud.

El Sistema Integral de Calidad en Salud SICALIDAD busca la integración de los proyectos de calidad, por eso propone:

1. Un sistema de calidad total: de la acreditación a la mejora continua, que garantice la continuidad de las iniciativas de calidad.
2. Incluir esfuerzos en todas las dimensiones de la calidad: técnica y seguridad del paciente, percibida y organizacional.
3. Entender la calidad como un atributo especial tanto en los programas de salud pública como en los de gestión de las enfermedades. La falta de integración de estos programas afecta a la eficiencia del sistema.
4. Integrar a todo el Sistema Nacional de Salud, lo que incluye al sector público, el sector privado y las organizaciones no lucrativas. Se trata de contar con una política nacional única de calidad en salud.
5. Finalmente, integrar proyectos referidos a la calidad, evitando acciones incomunicadas y aprovechando las sinergias de un modelo integrado de calidad total.

Y de acuerdo con la reforma a la Ley General de Salud en materia de Protección Social (DOF 15-05-2003, Art. 77 BIS 9) determina como un requisito indispensable que los establecimientos de salud que se incorporen al Sistema de Protección Social en Salud deben acreditar los requisitos mínimos de capacidad, calidad y seguridad para poder ofertar sus servicios y recibir el financiamiento correspondiente.

Con base al Reglamento Interior de la Secretaría de Salud (DOF 29-11-07 Cap. X, Art. 18), corresponde a la Dirección General de Calidad y Educación en Salud establecer, emitir y operar los instrumentos y mecanismos necesarios para el desarrollo del Sistema de Acreditación de establecimientos públicos de atención a la salud incorporados al Sistema de Protección Social en Salud.

SICALIDAD integra en su estrategia el proceso de acreditación como un mecanismo de garantía de calidad y de cumplimiento obligatorio para los prestadores del Sistema de Protección Social en Salud. Estos aspectos se señalan en el Manual para la Acreditación y Garantía de Calidad en Establecimientos para la Prestación de Servicios de Salud.

SICALIDAD indica la necesidad de supervisar periódicamente los establecimientos de salud acreditados como un mecanismo de mejora continua que garantice el mantenimiento de los requisitos de la acreditación y facilite la asociación gradual de estos establecimientos como el resto

de las líneas de acción. Se ha determinado documentar el impacto que representa el proceso de acreditación en los establecimientos de salud, respondiendo a un cuestionamiento lógico que determine lo que ha supuesto este proceso en términos de infraestructura, recursos y mejora de procesos.

Con base en estos aspectos SICALIDAD subdivide el proyecto de Acreditación y Garantía de Calidad en tres grandes grupos de acciones: el Sistema de Acreditación y Garantía de Calidad, la Supervisión de Establecimientos Acreditados y la Evaluación del Impacto de la Acreditación.

El establecer procedimientos para acreditar y supervisar los establecimientos de salud que se incorporen al Sistema de Protección Social en Salud, así como medir el impacto que supone el proceso de acreditación en dichos establecimientos permite el aseguramiento y garantía de la calidad.

De acuerdo a los procedimientos mínimos obligatorios se tienen que cumplir las Normas Oficiales Mexicanas que plasman el objetivo de describir los requerimientos necesarios de acuerdo a cada problemática de salud, nosotros señalaremos las referentes a la Prevención del Cáncer de Mama como NOM-041-SSA2-2011 y NOM-229-SSA1-2002 [SSA 06].

La norma NOM-229-SSA1-2002 [SSA 06], tiene como objetivo establecer los criterios de diseño, construcción y conservación de las instalaciones fijas y móviles, los requisitos técnicos para la adquisición y vigilancia del funcionamiento de los equipos de diagnóstico médico con rayos X, los requisitos sanitarios, criterios y requisitos de protección radiológica que deben cumplir los Titulares, responsables, Asesores Especializados en Seguridad Radiológica en establecimientos para el diagnóstico médico que utilicen equipos generadores de radiación ionizante (rayos X) para su aplicación en seres humanos, con el fin de garantizar la protección a pacientes, personal ocupacionalmente expuesto y público en general.

Es obligatorio observar a todos los propietarios el Territorio Nacional, así como Titulares, Responsables, Asesores Especializados en Seguridad Radiológica, equipos de rayos X y establecimientos para diagnóstico médico que utilicen equipos generadores de radiación ionizante (rayos X) en unidades fijas o móviles para su aplicación en seres humanos.

Establecer que todo el personal involucrado en el proceso de adquisición de imágenes mamográficas (médicos, técnicos, administrativos, etc.) observe lo establecido en un Programa de Garantía de Calidad, que de acuerdo a la Secretaría de Salud (SSA), es un conjunto de disposiciones administrativas, procedimientos técnicos, acciones para verificación y medidas correctivas, destinados a generar la máxima calidad de los servicios de diagnóstico médico con rayos X.

La importancia dentro de un Programa de Garantía de Calidad es el control de calidad de los equipos, lo que comprende su evaluación y la preservación de niveles óptimos de funcionamiento de todas sus componentes.

En un programa de control de calidad se establecen las pruebas a las que se deben someter los equipos, su frecuencia de aplicación, los criterios de aceptación que califican los resultados de las pruebas y las acciones que se deben tomar en caso de identificar fallos.

La norma NOM-041-SSA2-2011, por otro lado tiene como objetivo los criterios de operación para la prevención, diagnóstico, tratamiento, control y vigilancia epidemiológica del cáncer de mama. Siendo obligatoria para todo el personal de salud, profesional y auxiliar de los sectores público, social y privado que brinde atención médica. Esto permite mejorar el control en el manejo del Cáncer de Mama ya que actualmente es una de las principales causas de muerte de mujeres en México.

Por otro lado la Organización para la Normalización Internacional define el aseguramiento o la garantía de calidad como el conjunto de "todas las acciones planificadas y sistemáticas necesarias para inspirar suficiente confianza de que una estructura, sistema o componente va a funcionar a satisfacción cuando esté en servicio". Aplicando esta definición al radiodiagnóstico, la Organización Mundial de la Salud (OMS) (OMS, 1984) añade que "funcionar a satisfacción en servicio implica que pueda obtenerse la calidad óptima en todo el proceso de diagnóstico, es decir, que se produzca en todo momento una información de diagnóstico adecuada, y con una exposición mínima del paciente y del Personal".

Las Normas Internacionales Básicas de Seguridad (NBS) para la Protección contra la Radiación Ionizante y la Seguridad de las Fuentes de Radiación (IAEA, 1994) establecen el requisito de que toda práctica con radiación ionizante debe ser autorizada. Las instalaciones de radiología requieren, por tanto, autorización otorgada por la autoridad reguladora en materia de protección radiológica.

1. Establecen asimismo la responsabilidad principal para la aplicación de los requisitos de protección en el titular de la autorización y responsabilidades subsidiarias para el personal de las instalaciones, que en el caso de las instalaciones médicas incluye al personal médico y paramédico, y los expertos calificados.
2. En cuanto a las exposiciones médicas, dicho titular deberá cuidar de que "no se administre a ningún paciente una exposición médica con fines diagnósticos o terapéuticos a no ser que prescriba tal exposición un facultativo médico". La obligación global de la protección del paciente recae sobre los médicos, en el caso de los servicios de radiodiagnóstico, estos facultativos serían los médicos radiólogos.
3. En las aplicaciones diagnósticas de la radiación, los titulares de la autorización "deberían velar por que los requisitos sobre la formación de imágenes y garantía de calidad prescritos por las Normas se satisfagan con el asesoramiento de un experto cualificado en física de radiodiagnóstico o en física de medicina nuclear, según proceda".
4. Permiten flexibilidad suficiente para que cada país optimice su programa de acuerdo con la disponibilidad de expertos cualificados en física de radiodiagnóstico, ya que utiliza la expresión "debería", la cual tiene una flexibilidad inherente y sólo se pide que los aspectos de la formación de imagen y garantía de calidad se efectúen con el "asesoramiento" del experto calificado.

Entre los objetivos de los programas de aseguramiento de la calidad está lograr una calidad de imagen mamográfica alta, incluyendo todo el tejido mamario y a su vez los cinco elementos de calidad de imagen: contraste y resolución altas, ausencia de ruido, densidad ópticamedia y dosis por proyección; además, se debe incluir al posicionamiento de la mama como un elemento de calidad de la imagen. Otras características de los programas de calidad son:

1. Que los programas de aseguramiento de la calidad de la imagen sean equivalentes a los establecidos por el American College of Radiology (ACR), la Food and Drugs Administration (FDA) o el Organismo Internacional de Energía Atómica (OIEA).
2. Es fundamental la participación de un físico médico, el cual debe supervisar los programas de control de calidad en mamografía (aunque hagan los programas obligatorios de una forma muy básica, no equivalente a los normas del ACR, la FDA o el OIEA), y también los servicios de mamografía del país, en general.
3. Mediciones de los parámetros físicos de los generadores de radiación, los dispositivos de formación de imágenes y las instalaciones de irradiación en el momento de su puesta en servicio, supervisar periódicamente en lo sucesivo al personal experto calificado en físico de radiodiagnóstico.
4. La verificación de los factores físicos y clínicos apropiados utilizados para el diagnóstico o el tratamiento de los pacientes estén a cargo del personal de radiodiagnóstico.
5. Registros por escrito de los procedimientos significativos y sus resultados.

La importancia de la Prevención de Cáncer de Mama, es la detección temprana o diagnóstico, ya que al momento de hacerse una mamografía se debe exigir contar con equipo especializado y tecnología de punta, con personal médico y técnico altamente capacitado, así como con programas de garantía de calidad que aseguren la certeza del diagnóstico emitido, que la dosis de radiación impartida a la paciente sea mínima y la optimización de los costos del estudio. La mamografía de diagnóstico tiene también un gran impacto en la reducción de la mortalidad por este padecimiento, ya que un diagnóstico certero permite a las pacientes acceder a un tratamiento adecuado y así tener un mejor pronóstico.

El objetivo del **Manual de Control de Calidad en Mastografía** es sustituir la versión del 2002 y contribuir a las buenas prácticas hospitalarias, en el área de la una imagen de mama de acuerdo a la Norma Oficial Mexica NOM-041-SSA2-2011 [SSA 11], derivado de su apéndice normativo D, donde se mencionan las pruebas de control de calidad que se deben ejecutar por el personal Técnico Radiólogo a los mastógrafos analógicos y digitales. Con esta metodología se lograra la eficiente radiación o ionizaciones y obtener imágenes de alta calidad diagnóstica con el menor riesgo posible para el paciente y personal expuesto.(MCCM, 2012)

Las pruebas se dividen en "**Pruebas para equipos analógicos**" y "**Pruebas para equipos digitales**" donde se establecen los procedimientos, frecuencias y criterios de aceptación es decir se definen los protocolos.(MCCM, 2012)

Capítulo II – Marco Teórico

1. Software

Es un conjunto de instrucciones (programas de cómputo) que cuando se ejecutan proporcionan las características, funciones, desempeños buscados y la estructura de datos que permiten que los programas manipulen en forma adecuada la información, e información descriptiva tanto en papel como en formas virtuales que describen la operación y uso de los programas.

2. Ingeniería de Software

La *ingeniería de software* es un área de tecnología con varias capas como:

1. *Herramientas*
2. *Métodos*
3. *Procesos*
4. *Compromiso con la calidad*

*Las **herramientas*** son aquellos programas o paquetes que proporcionan un apoyo automatizado o semiautomático para el proceso y los métodos.

Los ***métodos*** proporcionan la experiencia técnica para elaborar software. Incluyen un conjunto amplio de tareas, como comunicación, análisis de los requerimientos, modelación del diseño, construcción del programa, pruebas y apoyo. También se basan en un conjunto de principios fundamentales que gobiernan cada área de la tecnología e incluyen actividades de modelación y otras técnicas descriptivas.

El ***proceso*** es una de las etapas más importante porque permite el desarrollo racional y oportuno del software de cómputo, forma la base para el control de la administración de proyectos, establece el contexto en el que se aplican métodos técnicos, se generan productos de trabajo, puntos de referencia, se asegura la calidad y se administra el cambio de manera apropiada.

Un ***proceso*** es un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto de trabajo. Una actividad busca logar un objetivo amplio. Una acción es un conjunto de tareas que producen un producto importante de trabajo. Una tarea se centra en un objetivo pequeño pero bien definido.

En el contexto de la ingeniería de software, un proceso no es una prescripción rígida de cómo elaborar software.

La ***estructura del proceso*** establece el fundamento para el proceso completo de la ingeniería de software por medio de la identificación de un número pequeño de actividades estructurales, que sean aplicables a todos los proyectos de software, sin importar su tamaño o complejidad. Una estructura de proceso general para la ingeniería de software consta de cinco actividades:

1. *Comunicación*: en esta actividad se buscan entender los objetivos de los participantes respecto al proyecto, y reunir los requerimientos que ayuden a definir las características y funciones del software que implica la intensa colaboración y comunicación con el cliente.
2. *Planeación*: en esta actividad se describen las tareas técnicas que deben realizarse, los recursos que serán requeridos, los productos de trabajo que han de producirse y un programa de las actividades, creando un "mapa" que guía al equipo.
3. *Modelado:* crea un "bosquejo" del software con el fin de entender el panorama general (cómo se verá arquitectónicamente, cómo ajustan entre sí las partes constituyentes y muchas características más).
4. *Construcción:* es una actividad que combina la generación de código y las pruebas que se requieren para descubrir errores en éste.
5. *Despliegue*: el software (como entidad completa o como un incremento parcialmente terminado) se entrega al cliente, el cual lo evalúa y que al mismo tiempo le da retroalimentación, misma que se basa en dicha evaluación.

Otras actividades relevantes son:

El seguimiento y control del proyecto de software: permite que el equipo de software evalúe el progreso comparándolo con el plan del proyecto y tome cualquier acción necesaria para apegarse a la programación de actividades.

Administración del riesgo: evalúa los riesgos que puedan afectar el resultado del proyecto o la calidad del producto.

Aseguramiento de la calidad del software: define y ejecuta las actividades requeridas para garantizar la calidad del software.

Revisiones técnicas: evalúa los productos del trabajo de la ingeniería de software.

Medición: define y reúne mediciones del proceso, proyecto y producto para ayudar al equipo a entregar el software.

Para comprender la *esencia práctica* de la ingeniería de software se requiere:

- Entender el problema (comunicación y análisis).
- Planear la solución (modelado y diseño del software).
- Ejecutar el plan (generación del código).
- Examinar la exactitud del resultado.

Los modelos de proceso de software intentan mejorar la calidad del sistema y hacer que los proyectos sean más manejables (en tiempo y costo), proporcionando una guía al equipo de software. Existen diferentes modelos de procesos de software, a continuación se explican los modelos evaluados para este proyecto.

3. Metodología en Cascada

Este modelo, algunas veces llamado el "ciclo de vida clásico", sugiere un enfoque sistemático y secuencial para el desarrollo de software. Inicia con la especificación de requerimientos del cliente y continúa con la planeación, el modelado, la construcción, y el despliegue para culminar en el soporte del software terminado.

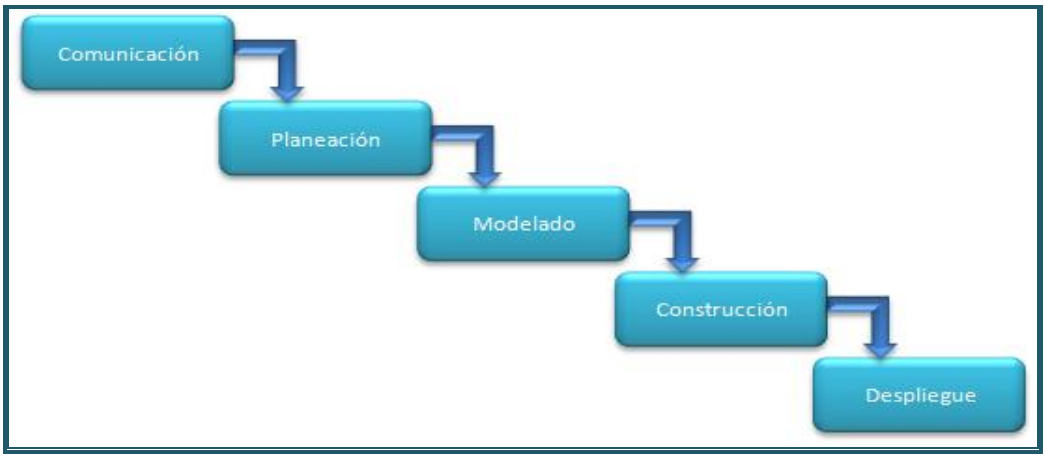


Figura 2.1 ***Modelado en cascada de acuerdo a (B., 1998).***

El ciclo de vida clásico consta de 7 fases:

1. ***Investigación preliminar***: esta etapa es crítica para el éxito del resto del proyecto, se requiere que el analista observe objetivamente los procesos que se desean automatizar. Los usuarios, los analistas y los administradores de sistemas que coordinan el proyecto son los involucrados en esta fase. Las actividades de esta fase consisten en entrevistar a los encargados de coordinar a los usuarios, sintetizar el conocimiento obtenido, estimar el alcance del proyecto y documentar los resultados.

2. ***Determinación de los requerimientos del sistema****:* esta etapa es útil para que el analista confirme la idea que tiene de la organización y sus objetivos, el analista se esfuerza por comprender la información que necesitan los usuarios para llevar a cabo sus actividades. Los implicados en esta fase son el analista y los usuarios, por lo general trabajadores y gerentes del área. Se necesitan conocer los detalles de los procesos de la empresa y se deben contestar las siguientes preguntas, ¿quién? (la gente involucrada), ¿qué? (la actividad de la empresa), ¿dónde? (el entorno donde se desarrollan las actividades), ¿cuándo? (el momento oportuno) y ¿cómo? (la manera en que se realizan los procedimientos actuales). El analista debe conocer el funcionamiento de la empresa y poseer información muy completa acerca de la gente, los objetivos, los datos y los procedimientos implicados.

3. ***Análisis de las necesidades del sistema***: en esta fase se realiza el análisis gráfico en diagramas de flujo de datos de las entradas, los procesos y las salidas de las funciones. A partir de los diagramas de flujo de datos se desarrolla un diccionario de los mismos,

permitiendo conocer todos aquellos que son utilizados en el sistema, así como sus respectivas especificaciones.

4. ***Diseño del sistema***: en esta fase el analista utiliza la información recopilada en las primeras fases para realizar el diseño lógico del sistema. La concepción de la interfaz de usuario forma parte del diseño lógico de este sistema. También incluye el diseño de archivos o de la base de datos. Una base de datos bien organizada es el cimiento de cualquier sistema de información. El analista también interactúa con los usuarios para diseñar las salidas de información, las cuales proveerán las herramientas necesarias para satisfacer los requerimientos de dichos usuarios.

5. ***Desarrollo y documentación del software****:* en esta fase el analista trabaja de manera conjunta con los programadores para desarrollar cualquier software original necesario. Durante esta fase el analista también trabaja con los usuarios para desarrollar documentación efectiva para el software, como lo son: manuales de procedimientos, ayuda en línea, sitios web, etc. La documentación indica a los usuarios como utilizar el software y lo que deben hacer en caso de que surjan problemas derivados de su uso. Los programadores desempeñan un rol clave porque diseñan, codifican y eliminan errores sintácticos de los programas de cómputo.

6. ***Prueba y mantenimiento del sistema***: Antes de poner en funcionamiento el sistema hay que probarlo. Es mucho menos costoso que encontrar los problemas antes que el sistema se entregue a los usuarios. Una parte de las pruebas las realizan los programadores solos, y otra la llevan a cabo de manera conjunta con los analistas. Primero se realizan una serie de pruebas con datos de muestra para determinar con precisión cuáles son los problemas y posteriormente se realiza otra con los datos reales del sistema actual. El mantenimiento del sistema de información y su documentación empiezan en esta fase. Gran parte del trabajo habitual del programador consiste en el mantenimiento, y las empresas invierten enormes sumas de dinero en esta actividad.

7. ***Implementación y evaluación del sistema****:* Esta es la última fase del desarrollo de sistemas, y aquí el analista participa en la implementación del sistema de información. En esta fase se capacita a los usuarios en el manejo del sistema. Parte de la capacitación la imparten los desarrolladores, pero la supervisión de ésta es responsabilidad del analista de sistemas. Además, el analista tiene que planear una conversión gradual de la actualización del sistema.

4. Metodología en Espiral

Es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistemáticos del modelo de cascada.

Es un generador de modelo de proceso, que se usa para guiar la ingeniería concurrente con participantes múltiples de sistemas intensivos en software. Tiene dos características principales. La primera es el enfoque cíclico para el crecimiento incremental del grado de un sistema y su

implementación, mientras que disminuye su grado de riesgo. La otra es un conjunto de puntos de referencia para asegurar el compromiso del participante con soluciones factible y mutuamente satisfactorias.

El software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, lo que se entrega puede ser un modelo o prototipo. En las interacciones posteriores se producen versiones cada vez más completas del sistema cuya ingeniería se está haciendo.

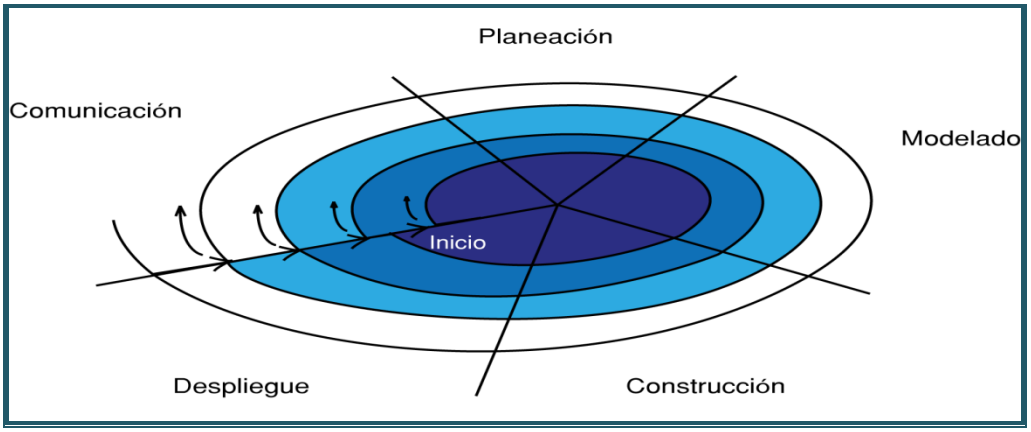


Figura 2.2 ***Modelado en espiral de acuerdo a (B., 1998).***

Un modelo en espiral es dividido por el equipo de software en un conjunto de actividades estructurales. Al comenzar el proceso evolutivo, el equipo de software realiza actividades implícitas en un circuito alrededor de la espiral en el sentido horario, partiendo del centro. El riesgo se considera conforme se desarrolla cada revolución. En cada paso evolutivo se marcan puntos de referencia puntales: combinación de productos del trabajo y condiciones que se encuentran a lo largo de la trayectoria de la espiral.

El primer circuito alrededor de la espiral da como resultado el desarrollo de una especificación del productor, las vueltas sucesivas se usan para desarrollar un prototipo y, luego, versiones cada vez más sofisticadas del software. Cada paso por la región de planeación da como resultado ajustes en el plan del proyecto. El costo y la programación de actividades se ajustan con base en la retroalimentación obtenida del cliente después de la entrega. Además, el gerente del proyecto ajusta el número planeado de interacciones que se requieren para terminar el software.

A diferencia de otros modelos, el modelo en espiral puede adaptarse para aplicarse a lo largo de toda la vida quizá del software. Entonces, el primer circuito alrededor de la espiral quizá represente un "un proyecto de desarrollo de producto nuevo". El nuevo producto evolucionara a través de cierto número de iteraciones alrededor de la espiral. Más adelante puede usarse un circuito alrededor de la espiral, cuando se caracteriza de este modo, sigue operativa hasta que el software se retira. Hay ocasiones en las que el proceso es inmóvil, pero siempre que se inicia un cambio comienza en el punto de entrada apropiado.

El modelo espiral es un enfoque realista para el desarrollo de sistemas y de software a gran escala. Como el software evolucionara a medida que el proceso avanza, el desarrollador y cliente

comprende y reaccionan mejor ante los riesgos en cada nivel de evolución. El modelo espiral usa los prototipos como mecanismo de reducción de riesgo, pero, más importante, permite aplicar el enfoque de hacer prototipos en cualquier etapa de la evolución del producto. El modelo espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y si se aplica de manera apropiada, debe reducir los riesgos antes de que se vuelvan un problema.

Pero, como otros paradigmas, el modelo espiral no es perfecto, en primer lugar es difícil convencer a los clientes de que el enfoque evolutivo es controlable, además demanda mucha experiencia en la evaluación del riesgo y se basa en ella para llegar al éxito.

5. Proceso Unificado

Es un modelo que analiza la necesidad de un proceso de software "impulsado por el caso de uso, centrado en la arquitectura, iterativo e incremental".

El Proceso Unificado (***PU***) tiene dos dimensiones:

1. Un eje horizontal que muestra el tiempo y muestra los aspectos del ciclo de vida del proceso a lo largo de su desenvolvimiento, que representa el aspecto dinámico del proceso conforme se va desarrollando y expresado en términos de fases, iteraciones e hitos (milestones).

2. Un eje vertical que muestra las disciplinas, las cuales se agrupan en actividades de una manera lógica de acuerdo a su naturaleza y representan el aspecto estático del proceso: en términos de componentes del proceso, disciplinas, actividades, flujos de trabajo, artefactos y roles.

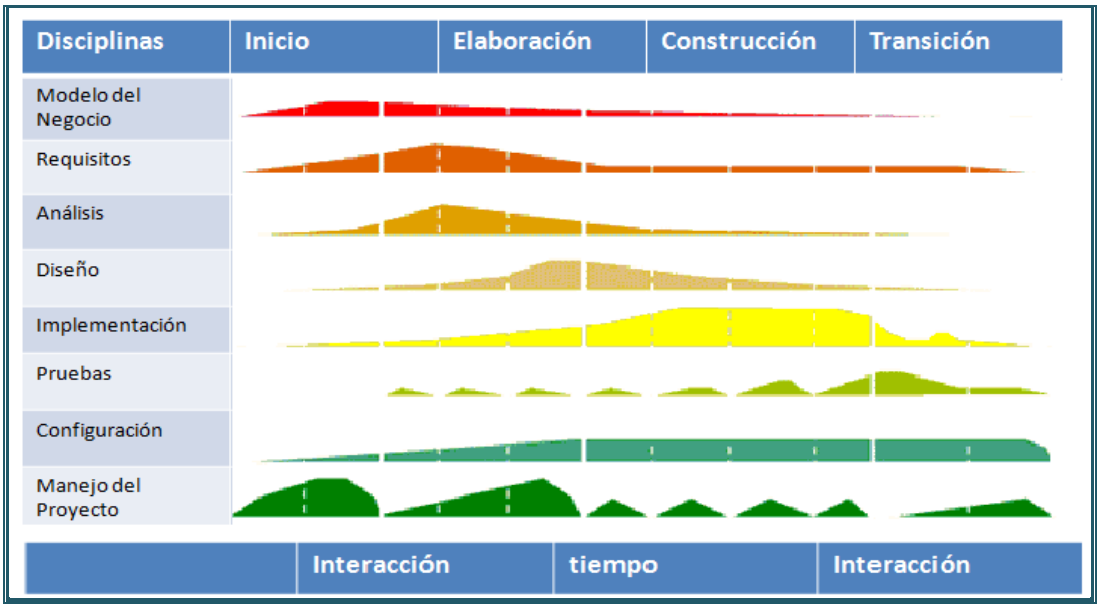


Figura2.3 ***Ciclo de vida del desarrollo de software del proceso unificado acuerdo a RUP.***

El Proceso Unificado se basa en componentes (*component-based*), lo que significa que el sistema en construcción está hecho de componentes de software interconectados por medio de interfaces bien definidas (*well-defined* interfaces). El Proceso Unificado utiliza *UML* (Lenguaje de Modelado Unificado) en la preparación de todos los planos del sistema de forma integral.

El Proceso Unificado está conformado por tres conceptos clave: dirigido por casos de uso (*use-case driven*), centrado en la arquitectura (*architecture-centric*), e iterativo-incremental.

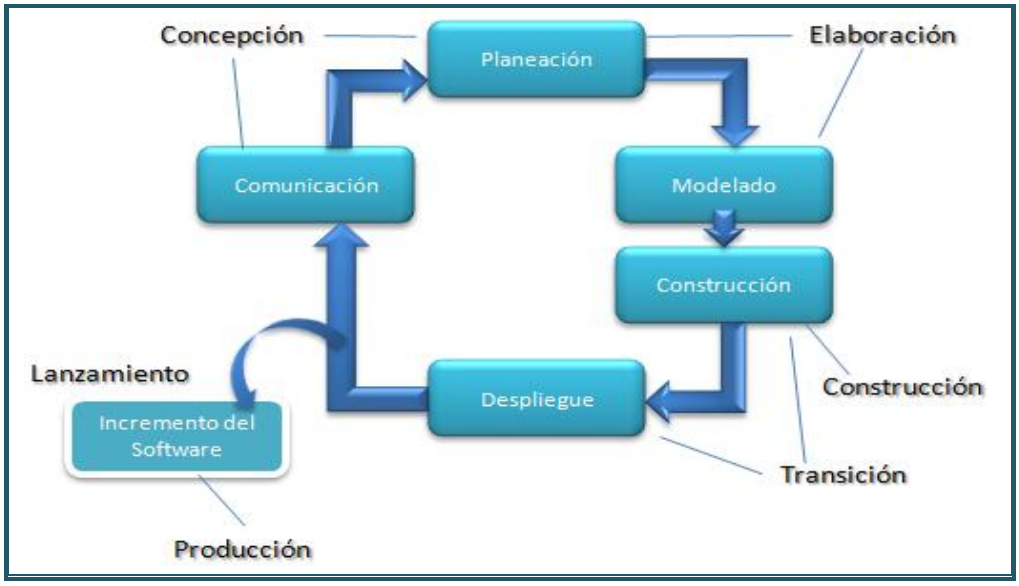


Figura2.4 ***Fases y flujo del proceso unificado.***

La ***fase de la concepción*** de *PU* agrupa actividades tanto de comunicación con el cliente como de planeación. Al colaborar con los participantes, se identifican los requerimientos del negocio, se propone una arquitectura aproximada para el sistema y se desarrolla un plan para la naturaleza iterativa e incremental del sistema. Los requerimientos fundamentales del negocio se describen por medio de un conjunto de casos de uso preliminares. La arquitectura se mejorara paulatinamente en un conjunto de modelos que representarán distintos puntos de vista del sistema. La planeación identifica los recursos, evalúa los riesgos principales, define un programa de actividades y establece una base para las fases que se van a aplicar a medida que avanza el incremento sistema.

La ***fase de elaboración*** incluye las actividades de comunicación y modelado general del proceso. La elaboración mejora y amplia los casos de uso preliminares desarrollados como parte de la fase de la concepción y aumenta la representación de la arquitectura para incluir cinco puntos de vista distintos del software: los modelos del caso de uso, de requerimientos, del diseño, de la implementación y del despliegue. En ciertos casos, la elaboración crea una "línea de base de la arquitectura ejecutable" que representa un sistema en su "primer corte". La línea de base de la arquitectura demuestra la viabilidad de ésta, pero no proporciona todas las características y funciones que se requieren para usar el sistema. Al término de la fase de elaboración se revisa con cuidado el plan a fin de asegurar el alcance, riesgos y fechas de entrega siguen siendo razonables. Es frecuente que en este momento se hagan modificaciones al plan.

*La **fase de construcción*** del *PU* es idéntica a la actividad de construcción definida para el proceso general del software. Con el uso del modelo de arquitectura como entrada, la fase de construcción desarrollar adquiere los componentes del software que harán que cada caso de uso sea operativo para los usuarios finales. Para lograrlo, se completan los modelos de requerimientos y diseño que se comenzaron durante la fase de elaboración, a fin de que reflejen la versión final del incremento de software. Después se implementan en código fuente todas las características y funciones necesarias para el incremento de software. A medida de que se implementan los componentes, se diseñan y efectúan pruebas unitarias para cada uno. Además, se realizan actividades de integración. Se emplean Caso de Uso para obtener un grupo de pruebas de aceptación que se ejecutan antes de comenzar la siguiente fase del *PU*.

*La **fase de transición*** del *PU* incluye las últimas etapas de la actividad general de construcción y la primera parte de la actividad de despliegue general (entrega y retroalimentación). Se da el software a los usuarios finales para la prueba beta, quienes reportan tanto los defectos como los cambios necesarios. Además, el equipo de software genera la información de apoyos necesaria que se requiere para el lanzamiento. Al finalizar la fase de transición, el software incrementado se convierte en un producto utilizable que se lanza.

*La **fase de producción*** del *PU* coincide con la actividad de despliegue del proceso general. Durante esta fase, se vigila el uso que se da al software, se brinda apoyo para el ambiente de operación (infraestructura) y se reportan defectos y solicitudes de cambio para su evaluación.

Es probable que al mismo tiempo que se lleve a cabo las fases de construcción, transición y producción, comience el trabajo sobre el siguiente incremento del software. Esto significa que las cinco fases del *PU* no ocurren en secuencia sino que concurren en forma escalonada.

El flujo de trabajo de la ingeniería de software está distribuido a través de todas las fases del *PU*. En el contexto de éste, un flujo de trabajo es análogo al conjunto de tareas. Es decir flujo de trabajo identifica las tareas necesarias para completar una acción importante de la ingeniería de software y los productos de trabajo que se generan como consecuencia de la terminación exitosa de aquéllas. Debe notarse que no toda tarea identificada para el flujo de trabajo del *PU* es realizada en todos los proyectos de software. El equipo adapta el proceso (acciones, tareas, subtareas y productos del trabajo) a fin de que cumpla sus necesidades.

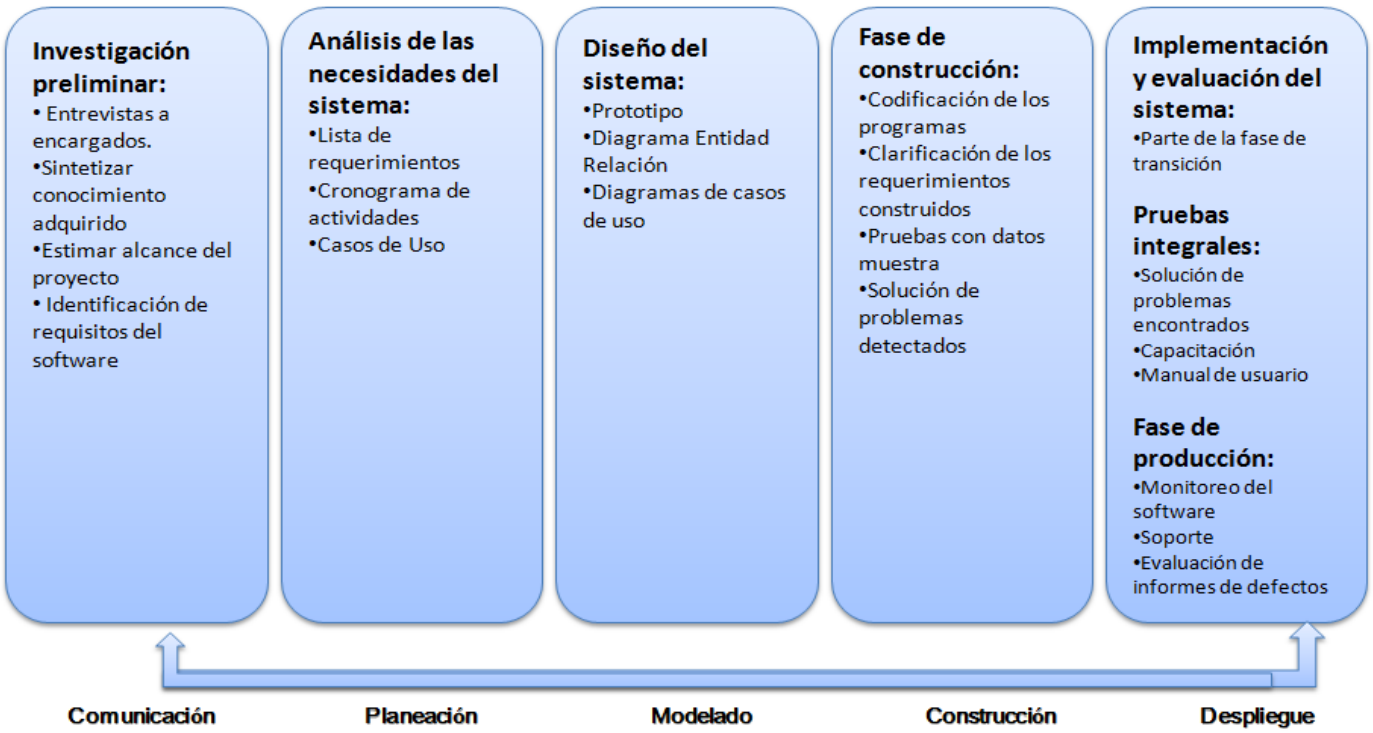
6. Metodología implementada

METODOLOGÍA DE CASCADA

METODOLOGÍA DE ESPIRAL

PROCESO UNIFICADO

METODOLOGÍA EMPLEADA



7. Conceptos generales

7.1 Framework

Es un conjunto estandarizado de conceptos, prácticas y criterios para resolver un tipo de problemática particular, que sirve como referencia para enfrentar y solucionar problemas de índole similar, esto es una plataforma, entorno y marco de trabajo. Algunas de sus características son:

1. Facilitar el desarrollo de software.
2. Evitar los detalles de bajo nivel, permitiendo concentrar más esfuerzo y tiempo en identificar los requerimientos de software.
3. Siendo la extensión de un lenguaje mediante una o más jerarquías de clases que implementan una funcionalidad y que pueden ser extendidas.

Es una estructura de soporte definida, en la cual un proyecto de software puede ser organizado y desarrollado, suele incluir:

1. Soporte de programas.
2. Bibliotecas (*TagLibraries*).
3. Software para desarrollar y unir diferentes componentes de un proyecto, como desarrollo de programas (*Scriptlets*).

7.2 Integrated Development Environment (IDE)

IDE (Integrated Development Environment, *IDE*), es un conjunto de herramientas de programación es decir un entorno de programación entre sus características incluye:

1. Editor de texto.
2. Utiliza un compilador-interprete.
3. Posee un depurador.
4. Manejo de un sistema de control de versiones.
5. Ayuda para interfaces de usuario.
6. Automatiza tareas (ANT, Maven, etc).

Algunos ejemplos de *IDE* son: Eclipse, NetBeans, etc.

7.3 Modelo Vista Controlador (MVC)

MVC (Model View Controler, *MVC*) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

1. Modelo (Objetos de Negocio): Es una representación específica de toda la información con la cual el sistema va a trabajar. La lógica de datos puede llegar a asegurar la integridad de ellos y permite derivar nuevos datos.
2. Vista (Interfaz con el usuario u otros sistemas): Es la representación de un modelo con el que interactúa el usuario.
3. Controlador (Controlador del *workflow* de la aplicación): El controlador responde a eventos, ó acciones que el usuario invoca, que implican cambios en el modelo y también en la vista (interfaz).

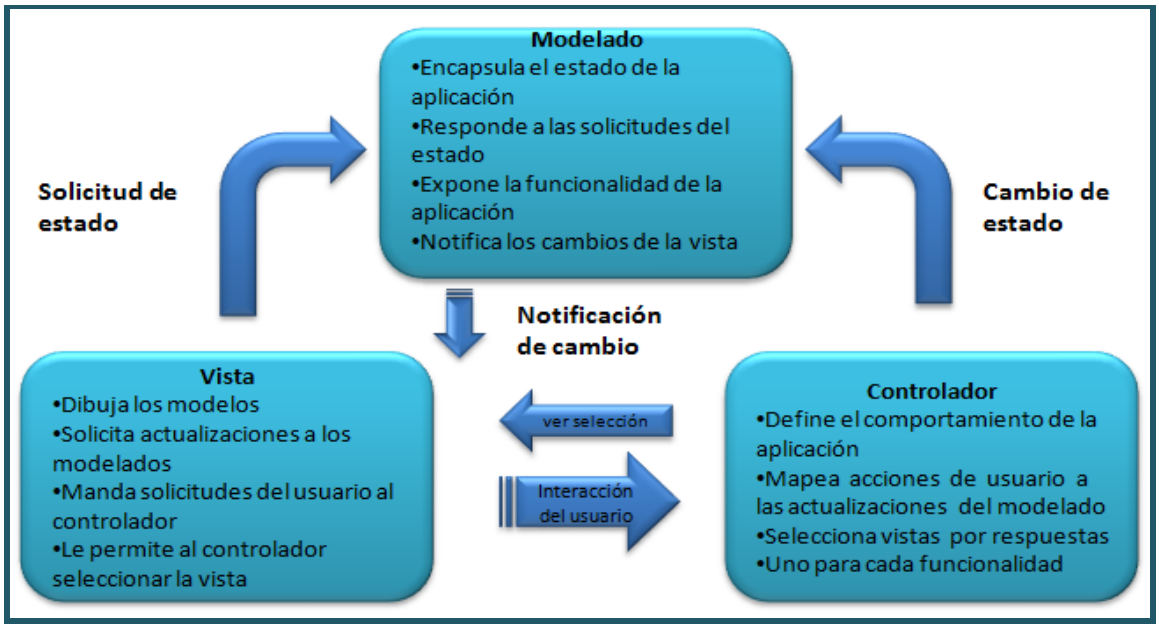


Figura 2.5 ***Patrón de diseño del modelo vista controlador.***

7.4 Unified Modeling Language (UML)

El Lenguaje Unificado de Modelado (Unified Modeling Language, *UML*) es un lenguaje estándar para escribir planos de software. *UML* puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucren una gran cantidad de software.

UML es sólo un lenguaje y, por tanto, es tan sólo una parte de un método de desarrollo de software, es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

Como es un lenguaje proporciona un vocabulario y reglas para combinar palabras de ese vocabulario con el objetivo de posibilitar la comunicación. Un lenguaje de modelado es un lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema. Es un lenguaje estándar para los planos del software. El modelado proporciona una comprensión de un sistema. El vocabulario y las reglas de un lenguaje como *UML* indican cómo crear y leer modelos bien formados, pero no indica cuales ni cuándo.(UML, 2009)

UML es algo más que un simple montón de símbolos gráficos. Detrás de cada símbolo en la notación *UML* hay una semántica bien definida. De esta manera, un desarrollador puede escribir un modelo en *UML*, y otro desarrollador, o incluso otra herramienta, puede interpretar ese modelo sin ambigüedad.

Especificar significa construir modelos precisos, no ambiguos y completos. La especificación de todas las decisiones de análisis, diseño e implementación que deben realizarse al desarrollar y desplegar un sistema. *UML* es un lenguaje para documentar, cubre la documentación de la arquitectura de un sistema y todos sus detalles. *UML* también proporciona un lenguaje para expresar requisitos y pruebas además proporciona un lenguaje para modelar las actividades de planificación de proyectos y gestión de versiones.

El vocabulario de *UML* incluye tres clases de bloque básicos:

1. Elementos
2. Relaciones
3. Diagramas

Los elementos son abstracciones que constituyen es lo más importante de una clase en un modelo; las relaciones ligan estos elementos entre sí; los diagramas agrupan colecciones interesantes de elementos.

Tipos de elementos en *UML*

1. Elementos estructurales.
2. Elementos de comportamiento.
3. Elementos de agrupación.
4. Elementos de anotación.

Elementos Estructurales son los nombres de modelos *UML*. En su mayoría son las partes estáticas de un modelado, y representan conceptos o cosas materiales. Colectivamente, los elementos estructurales se denominan clasificadores.

Una ***clase*** es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

Una ***interfaz*** es una colección de operaciones que especifican un servicio de una clase o componente. Una interfaz puede representar el comportamiento completo de una clase o componente o sólo una parte de ese comportamiento.

Una ***colaboración*** define una interacción y es una sociedad de roles y otros elementos que colaboran para proporcionar un comportamiento cooperativo mayor que la suma de los comportamientos de sus elementos.

Un ***caso de uso*** es una descripción de secuencias de acciones que ejecuta un sistema y produce un resultado observable de interés para un actor particular.

Una ***clase activa*** es una clase cuyos objetos tienen uno o más procesos o hilos de ejecución y, por tanto, pueden dar origen a actividades de control.

Un ***componente*** es una parte modular del diseño del sistema que oculta su implementación tras un conjunto de interfaces externas.

Un ***artefacto*** es una parte física y reemplazable de un sistema que contiene información física ("bits"). En un sistema hay diferentes tipos de artefactos de despliegue, como archivos de código fuente, ejecutables y scripts. Un artefacto representa típicamente el empaquetamiento físico de código fuente o información en tiempo de ejecución.

Los ***elementos de comportamiento*** son las partes dinámicas de los modelos *UML*. Éstos son los verbos de un modelo, y representan comportamiento en el tiempo y espacio. Hay tres tipos de comportamiento.

1. Una *interacción* es un comportamiento que comprende un conjunto de mensajes intercambios entre un conjunto de objetos, dentro de un contexto particular, para alcanzar un propósito específico.
2. Una *máquina* de estados es un comportamiento que especifica las secuencias de estados por las que pasa un objeto o una interacción durante su vida en respuesta a eventos, junto con sus reacciones a estos eventos.
3. Una *actividad* es un comportamiento que especifica la secuencia de pasos que ejecuta un proceso computacional.

Los ***elementos de agrupación*** son las partes organizativas de los modelos UML. Estos son las cajas en las que se puede descomponerse un modelo. Hay un tipo principal de elementos de agrupación: los paquetes.

Un ***paquete*** es un mecanismo de propósito general para organizar el propio diseño, en oposición a las clases, que organizan construcciones de implementación.

Los ***elementos de anotaciones*** son las partes explicativas de los modelos UML. Son comentarios que se pueden aplicar para describir, clasificar y hacer observaciones sobre cualquier elemento de un modelo.

Relaciones en *UML*

1. Dependencia.
2. Asociación.
3. Generalización.
4. Realización.

La ***dependencia*** es una relación semántica entre dos elementos, en la cual un cambio a un elemento (el elemento independiente) puede afectar a la semántica del otro elemento (el elemento dependiente).

Asociación es una relación estructural entre clases que describe un conjunto de enlaces, los cuales son conexiones entre objetos que son instancias de clases.

Generalización es una relación de especialización/generalización en la cual el elemento especializado (el hijo) se basa en la especificación del elemento generalizado (el padre).

Realización es una relación semántica entre clasificadores en donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá.

Diagrama en *UML*. Es la representación gráfica de un conjunto de elementos, visualizando la mayoría de las veces como un grafo conexo de nodos (elementos) y arcos (relaciones). Los diagramas se dibujan para visualizar un sistema desde diferentes perspectivas, de forma que un diagrama es una proyección de un sistema.

1. Diagrama de clases
2. Diagrama de objetos
3. Diagrama de componentes
4. Diagrama de estructura compuesta
5. Diagrama de caso de uso
6. Diagrama de secuencia
7. Diagrama de comunicación
8. Diagrama de estados
9. Diagrama de actividades
10. Diagrama de despliegue
11. Diagrama de paquetes
12. Diagrama de tiempos
13. Diagrama de visión global de interacciones

Diagrama de clase muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Estos diagramas son los más comunes en el modelado de sistemas orientados a objetos.

Diagrama de objetos muestra un conjunto de objetos y sus relaciones. Los diagramas de objetos representan instantáneas estáticas de instancias de los elementos existentes en los diagramas de clases.

Diagrama de componentes representa la encapsulación de una clase, junto con sus interfaces, puertos y estructura interna, la cual está formada por otros componentes anidados y conectores.

Diagrama de caso de uso muestra un conjunto de caso de uso, actores y relaciones.

Diagrama de estados muestra una máquina de estados, que consta de estados, transiciones, eventos y actividades, muestra la vista dinámica de un objeto.

Diagrama de actividades muestra la estructura de un proceso u otra computación como el flujo de control y datos paso a paso en la computación.

Diagrama de despliegue muestra la configuración de nodos de procedimientos en un tiempo de ejecución y los artefactos que residen en ellos. Abordan la vista de despliegue estática de una arquitectura.

Diagrama de artefactos muestra los constituyentes físicos de un sistema en el computador.

Diagrama de paquetes muestra la descomposición del propio modelo en unidades organizativas y sus dependencias.

Diagrama de tiempos es un diagrama de interacción que muestra los tiempos reales entre diferentes objetos o roles, en oposición a la simple secuencia relativa de mensajes.

Reglas de *UML* sintácticas y semánticas para:

1. Nombres: como llamar a los elementos, relaciones y diagramas.
2. Alcance: el contexto que da un significado específico a un nombre.
3. Visibilidad: cómo se pueden ver y utilizar esos nombres por otros.
4. Integridad: cómo se relacionan apropiada y consistentemente unos elementos con otros.
5. Ejecución: que significa ejecutar o simular un modelo dinámico.

Los modelos construidos durante el desarrollo de un sistema con gran cantidad de software tienden a evolucionar y pueden ser vistos por muchos usuarios de formas diferentes y en momentos diferentes.

7.5 Action

Es el mediador entre la Vista y el Modelo, ya que transmite los datos desde la Vista hacia el proceso específico del modelo y retorna los resultados en sentido contrario. Un Action funciona como un adaptador entre una petición *HTTP* entrante y la lógica de negocio creando una instancia de la *Action* correspondiente y ejecutando el método *execute()*.

1. Por lo tanto es necesario que la subclase sobreescriba el método *execute()*.
2. No se debe añadir lógica de negocio en ella, mejor utilizar otra capa *DAO*.
3. Método *execute*():

Action class

La clase *Action* contiene la lógica de negocio, encargada de recuperar el paquete de recursos, mantener los datos, validar y seleccionar los resultados de la vista que se deben enviar de vuelta al usuario. Es el corazón del *Struts* 2.

1. Invocar adecuadamente la lógica de negocios.
2. *Data-accesslogic*.
3. Almacenar los resultados en *beans*.
4. Diseñar el tipo de situación (datos faltantes, errores de base de datos, etc).
5. Sus parámetros son:
    a. *mapping* -de *ActionMapping* usado para seleccionar esta instancia.
    b. *form* - ActionForm bean para esta petición.
    c. *request* – es la petición *HTTP* que se está procesando.
    d. *response* – es la respuesta *HTTP* que se está creando.
6. Lanza una excepción si en la lógica de negocio ocurre también una excepción.

ActionSupport

La clase *Support*, es práctica común para proporcionar implementaciones por defecto de interfaces.

El *ActionSupport* es una clase muy potente y útil que proporciona implementación predeterminada de algunas de las interfaces importantes.

La clase *ActionSupport* tiene la capacidad de hacer:

1. Validaciones - Declarando un método *validate*() y poner el código de validación en el interior.
2. Localización de texto-utilice el método *GetText*() para obtener el mensaje de paquete de recursos.

7.6 Base de Datos Relacional

Una base de datos relacional es una base de datos en donde todos los datos son visibles al usuario y están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos se realizan sobre estas tablas.

Estas bases de datos son percibidas por los usuarios como una colección de relaciones normalizadas en diversos grados que varían con el tiempo.

El modelo relacional representa un sistema de bases de datos en un nivel de abstracción alejado de los detalles de la máquina subyacente, de la misma forma como, un lenguaje del tipo de PL/1 representa un sistema de programación con un nivel de abstracción y también alejado de los detalles

de la máquina subyacente. De hecho, el modelo relacional puede considerarse como un lenguaje de programación abstracto, orientado hacia las aplicaciones de bases de datos (Date,1993).

En el modelo relacional se deben considerar el diseño conceptual y lógico:

Diseño Conceptual, cuyo objetivo es obtener una representación de la información con independencia de usuarios y aplicaciones en particular, y fuera de consideraciones sobre la eficiencia del ordenador.

Diseño Lógico, cuyo objetivo es transformar el diseño conceptual obtenido y adaptarlo al modelo de datos en el que se apoya el *SGBD* (Sistema de Gestión de Base de Datos) que se va a utilizar.

Además este modelo relacional debe considerar la cardinalidad.

La ***Cardinalidad*** se da con base a las posibilidades de relación entre las entidades, existiendo tres tipos de cardinalidad:

1. Cardinalidad 1:1, es cuando una entidad *A* se relaciona solo con otra entidad *B* y viceversa. Por ejemplo, el identificador de un coche (número de bastidor) se corresponde con una matrícula y esa matrícula con ese identificador del coche.

2. Cardinalidad *1:N*, es cuando una entidad *A* se puede relacionar con *N* entidades *B* pero no al revés. Por ejemplo, un libro puede tener *N* ejemplares, pero un ejemplar es solo de un libro.

3. Cardinalidad *N:M*, es cuando una entidad *A* se relaciona con *N* entidades *B* y viceversa. Por ejemplo, un libro puede ser escrito por varios autores distintos y un autor puede escribir varios libros distintos.

La conversión del diseño conceptual al diseño lógico está basada en los tres principios siguientes:

1. Todo tipo de entidad del modelo conceptual se convierte en una tabla.
2. Todo tipo de relación entre tablas *1:N* se traduce en una propagación de la clave (se crea una clave primaria o foránea) o bien se crea una nueva tabla intermedia.
3. Todo tipo de relaciones entre tablas *N:M* (muchos a muchos) origina la creación de una nueva tabla intermedia.

Teoría de la normalización.

En el desarrollo del diseño lógico obtenemos una serie de tablas finales que son las candidatas a formar nuestra base de datos. Sin embargo, dichas tablas han sido obtenidas a partir de un diseño

conceptual elaborado sin ningún tipo de reglas, por lo que podemos obtener un diseño de tablas más heterogéneo.

La teoría de la normalización consiste en un conjunto de reglas formales que nos permiten asegurar que un diseño lógico cumple una serie de propiedades, corrigiendo la estructura de los datos, de las tablas y evitando una serie de problemas como:

1. Incapacidad de almacenar ciertos hechos.
2. Redundancias y, por tanto, posibilidad de inconsistencias.
3. Ambigüedades.
4. Pérdida de información.
5. Aparición en la base de datos de estados no válidos en el mundo real, es lo que se llama anomalías de inserción, borrado y modificación.

Las reglas formales de la teoría de la normalización son conocidas con el nombre de formas normales. Existen seis formas normales, de manera que cuando la base de datos cumple las reglas de la primera forma normal se considera que está en primera forma normal (*1FN*), cuando pasan la segunda, que está en segunda forma normal (*2FN*), etc. Además, una base de datos de la que se afirme que está en *2FN*, está también en *1FN*, pues las formas normales se aplican de forma sucesiva.

De las seis formas normales, generalmente solo se aplican sobre las bases de datos las tres primeras, considerando que una base de datos que está en *3FN* es una base de datos correctamente diseñada. Expondremos a continuación algunas formas normales.

Primera forma normal (1FN).

Una base de datos se considera que está en *1FN* si cada atributo (campo) de una tabla contiene un solo valor atómico (simple). Un atributo que contiene varios valores puede derivar en una pérdida de datos.

Segunda forma normal (2FN).

La segunda forma normal, se relaciona con el concepto de dependencia funcional. Entendemos como dependencia funcional a la relación que tienen los atributos (campos) de una tabla con otros atributos de la propia tabla. Un campo tiene dependencia funcional si necesita información de otro/s campo/s para poder contener un valor.

Una tabla se dice que está en segunda forma normal (*2FN*) si sucede que:

1. Está en *1FN*.
2. Cada atributo (campo) no clave depende de la clave completa, y no parte de ella.

Por supuesto, una base de datos estará en *2FN* si todas sus tablas lo están. La idea intuitiva de la *2FN* es identificar todas las tablas con una clave compuesta, pues todas las tablas con clave simple están por defecto en *2FN* si están en *1FN*, y comprobar que cada uno de los campos de esta tabla depende de la clave completa.

Tercera forma normal (3FN).

Una tabla se dice que está en tercera forma normal (*3FN*) si:

1. Está en *2FN*.
2. Todos los atributos que no son claves deben ser mutuamente independientes, es decir, un atributo no debe depender de otro atributo no clave de su tabla.

Si un atributo que no es clave depende de otro atributo que no es clave, la tabla posiblemente contiene datos acerca de más de una entidad, contradiciendo el principio de que cada tabla almacene información de una entidad.

Obteniendo un diseño conceptual que después es convertido en diseño lógico, este diseño lógico resultante estará en *3FN* siempre que todo el proceso se haya realizado de forma correcta, sirviendo en este caso la teoría de la normalización para comprobar que el diseño ha sido realizado correctamente. Si no lo fuese, podremos aplicar las formas normales para corregir los errores que hubieran podido producirse.

Mientras la normalización resuelve problemas relacionados con la estructuración de los datos en tablas, crea problemas añadidos a su propio concepto, como son la duplicación de datos y la ineficacia en la recuperación de información.

Así, el proceso de normalización envuelve la descomposición de una tabla en tablas más pequeñas, lo cual requiere que la clave primaria de la tabla original se incluya, como una clave foránea, en la tabla/s que se forman. Esto significa que a medida que se van creando estas claves foráneas se va incrementando las probabilidades de poner en peligro la integridad de la base de datos.

La cuarta y quinta formas normales tratan con campos que pueden tener diferentes valores. Un campo que puede tener diferentes valores debe corresponder a una relación muchos a muchos ó muchos a uno (*N:M, M:1*). En ese sentido la cuarta y la quinta forma normal son también llaves compuestas. Estas llaves normales intentan minimizar el número de campos involucrados en la llave compuesta.

Cuarta forma normal (4FN).

Un registro no debe contener campos que acepten diferentes valores independientes en una entidad y el registro debe satisfacer la tercera forma normal. De no cumplir con esta forma normal se cae en incertidumbre. Las políticas de mantenimiento son posibles en dos campos que aceptan diferentes

valores independientes en un registro. Si hay repeticiones, entonces la actualización debe hacerse en múltiples registros, y los registros pueden volverse inconsistentes.

Dependencias multivaluadas son definidas esencialmente como una relación que acepta la política de mantenimiento de él producto cruzado. La dependencia multivaluada y la *4FN* también aplica a las relaciones que involucran a más de dos campos.

Quinta forma normal (5FN).

La quinta forma normal trata con casos donde la información puede ser reconstruida de piezas pequeñas de información que puede ser mantenida con menos redundancia. La segunda, tercera y cuarta formas normales también sirven a este propósito, pero la quinta forma normal generaliza los casos no cubiertos por las otras.

Podemos decir que un registro si cumple con la *5FN* cuando la información que contiene no puede ser reconstruida de varios registros pequeños (de registros que cada uno tienen menos campos que el registro original).

El caso donde todos los registros pequeños tienen la misma llave es excluido. Si un registro puede ser solo descompuesto en registros pequeños donde todos tienen la misma llave, entonces el registro es considerado dentro de la *5FN* sin descomposición. Un registro que cumple con la *5FN* cumple también con la cuarta, tercera, segunda y primera forma normal.

La *5FN* no difiere de la *4FN* a menos de que exista una limitación simétrica. Una ventaja de la *5FN* es que ciertas redundancias pueden ser eliminadas. Se observa que la normalización involucra más registros, deben existir un total menor de ocurrencias.

La ventaja es notoria cuando se almacenan un numero grandes de registros, mientras que las bases de datos normalizadas crecen en forma sumatoria, las bases de datos no normalizadas crecen en forma multiplicativa.

7.7 Extensible Markup Languaje (XML)

XML (Extensible Markup Language, *XML*) es un lenguaje de marcas muy similar a *HTML*, fue diseñado para el transporte de datos, no para mostrar los datos. Las etiquetas XML no están predefinidas. Se deben definir sus propias etiquetas además está diseñado para ser auto-descriptivo. *XML* es un complemento de *HTML*.

7.8 Object-Graph Navigation Languaje (OGNL)

OGNL (Object-Graph Navigation Language, *OGNL*) creado por la tecnología *OGNL*, se basa en el lenguaje de programación Java, que, durante el uso de expresiones más simples de una gama completa, admitidos por el lenguaje Java, permite obtener y establecer propiedades (a través de los

métodos definidos *setProperty* y *getProperty*, que se encuentran en *JavaBeans*), y la ejecución de los métodos de clases de Java. También permite la manipulación de matrices simples.

Las expresiones *OGNL* son útiles para la evaluación y manipulación de los valores de atributo, así como para devolver información sobre la base de los resultados. También puede transformar un rango de valores en una descripción de texto, o hacer lo mismo para una secuencia de intervalos.

OGNL es la interfaz entre el marco *Struts* 2 basado en cadenas de entrada *HTTP* y salida además el procesamiento interno basado en Java. Con la ayuda de *OGNL*, el marco *Struts* 2 permite la transferencia de datos en más complejos secundarios en Java como tipos de listas, mapas, etc. Usando convertidores también los desarrollados pueden extender el mecanismo de conversión de tipos y puede manejar cualquier tipo de datos, incluyendo usuario tipos definidos.

Un framework utiliza un contexto de nomenclatura estándar para la evaluar expresiones *OGNL*. El framework establece el contexto *OGNL* es nuestro *ActionContext*, y la pila de valores a ser el objeto raíz *OGNL* (la pila de valor es un conjunto de varios objetos, sino para *OGNL* parece ser un único objeto). Junto con la pila de valores, los lugares marco de otros objetos en el *ActionContext*, incluyendo mapas que representan la aplicación, la sesión y contextos solicitud. Estos objetos conviven en el *ActionContext*, junto a la pila de valores (la raíz *OGNL*).

7.9 Patrones de Diseño

Los patrones de diseño son un conjunto de estrategias, o buenas prácticas, que pueden facilitar el trabajo en aplicaciones orientadas a objetos. Los patrones de diseño son independientes del lenguaje en el que se utilicen (siempre y cuando el lenguaje sea orientado a objetos). Generalmente se presentan como diagramas de *UML*.

Según su enfoque los patrones de diseño se agrupan en:

1. Patrones de creación (o creacionales): abstraen el proceso de la instanciación y ocultan detalles de cómo los objetos son creado o iniciados.
2. Patrones estructurales: describen como las clases y objetos pueden ser combinados para formas grandes estructuras y proporcionar nuevas funcionales.
3. Patrones de comportamiento: tratan del cómo definir la comunicación e interacción entre los objetos de un sistema con la finalidad de reducir el acoplamiento entre estos.
4. Patrones de sistema.

7.10 Garbage Collection(GC)

Recolector de basura (Garbage Collection, *GC*), es una rutina que busca en la memoria segmentos de programas o de datos que ya no son activos, con el fin de recuperar ese espacio.

En cada ocasión que se cree un objeto va a tener memoria en uso, así podríamos llenar el *heap* (esta es la zona de la memoria *dinámica*, los objetos son creados, eliminados o modificados en esta parte de la memoria), incluso podríamos llegar a tener un *OutOfMemoryException* (sí nos quedamos sin memoria, y si no se sabe utilizar), para eso necesitamos borrar los objetos que no tengan referencias, o que queden perdidos en memoria, en el lenguaje de programación C, la liberación de memoria está a cargo del programador, cuidando y eliminando manualmente los objetos que no se utilizan, pero Java tiene el *Garbage Collector*, que es un proceso de la *JVM* que está revisando que objetos pueden ser borrados y cuáles no.

¿Pero cómo la *JVM* sabe que objetos deben borrarse? Sencillo, si los objetos están perdidos en memoria, no tienen referencias, y por lo tanto no se pueden accesar, esos son los candidatos a ser borrados, en la siguiente pasada del *GC*.

El *GC* es un proceso de baja prioridad, por lo que no se pasa en todo momento liberando memoria, si no que pasa de vez en cuando, porque podría ser en un tiempo muerto del procesador, además nosotros podríamos sugerirle que pase, pero va a pasar cuando pueda y quiera, a continuación los siguientes ejemplos:

```
1 System.gc();
2 Runtime.getRuntime().gc();
```

Un ejemplo de *GC*

```
1 class MyClass {
2     public static void main(String[] args) {
3         MyClass myFirstObject = new MyClass();
4         MyClass mySecondObject = new MyClass();
5         myFirstObject = mySecondObject;
6         mySecondObject = null;
7     }
8 }
```

Línea 3. Se crea el primer objeto en el *heap*, se crea la primera referencia en el *stack* (podríamos imaginarlo como si fuese un *Array*, que es de tamaño fijo durante el tiempo de ejecución del programa, en tiempo de compilación es cuando se define el tamaño que tendrá), y la referencia apunta al objeto.

Línea 4. Se crea el segundo objeto en el *heap*, se crea la segunda referencia en el *stack*, y la referencia apunta al objeto.

Línea 5. La primera referencia apunta hacia el segundo objeto. En este momento el primer objeto se queda sin referencia, es desechado por el *GC*, y se libera memoria.

Línea 6. La segunda referencia, apunta a nada.

7.11 *Inyección de Dependencias(DI)*

Inyección de Dependencias (Dependency Injection, *DI*), es una herramienta comúnmente utilizada en varios patrones de diseño orientado a objetos, consiste en inyectar comportamientos a componentes.

Esto no es más que extraer responsabilidades a un componente para delegarlas en otro, estableciendo un mecanismo a través del cual el nuevo componente pueda ser cambiado en tiempo de ejecución.

La Inyección de Dependencias, es colocar dentro de un objeto otros que puedan cambiar su comportamiento, sin que esto implique volver a crear el objeto.

Esto nos permite tener un objeto que puede hacer un conjunto de tareas, cada una de esas tareas es una responsabilidad, que puede ser ejecutada por otro objeto únicamente especialista y dedicado a ello (una responsabilidad), pero ahora tenemos otro diferenciador.

1. El objeto responsable de ejecutar esa única tarea.
2. Puede establecerse en tiempo de ejecución.

La implementación habitual en programación es crear un método capaz de establecer el comportamiento, es decir capaz de cambiar el valor de un atributo asignándole una instancia de objeto diferente.

7.12 Lenguajes de Expresiones (EL)

JSTL define un Lenguaje de Expresiones (*EL*), que se basa en expresiones regulares para facilitar el tratamiento de información, cuya sintaxis es: ${expresión}.

En las expresiones se usan los operadores +,-,*,/,mod, >,<,<=,>=,==,!=, &&, ||, ! y el operador *empty* para comparar con *null* y con cadena vacía.

Con *EL* accedemos a todos los objetos implícitos de *JSP*, y se añaden los objetos *param*, *paramValues* y *header*. En *EL* los objetos implícitos disponibles son: *pageContext*, *pageScope*, *requestScope*, *sessionScope*, *applicationScope, param, paramValues* y *header*.

Para acceder a un atributo dentro de un objeto, podemos usar los operadores '.' y '[]', de la siguiente forma:

- objeto.atributo
- objeto ["nombreAtributo"].

7.13 Object Relational Mapping (ORM)

El Mapeo Objeto-Relacional (Object-Relational Mapping, o sus siglas *O/RM, ORM*, y *O/R mapping*) es una técnica de programación para convertir una base de datos relacional a un lenguaje de programación orientado a objetos. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional.

Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo).

Las bases de datos relaciones guardan datos primitivos, por esta razón los datos del objeto se convierten en datos primitivos que se almacenan en las tablas de una base de datos. Y como se requiriera el objeto en la aplicación se vuelve a construir el mismo, para recuperar los datos primitivos de la base de datos. Con el *ORM* nos olvidados de cómo convertir los objetos en datos primitivos para almacenarlos y viceversa.

Se puede relacionar de la siguiente forma la, *Table gateway* y la *Table-row gateway*:

Programa OO	**BD Relacional**
Clase	Tabla
Propiedad	Campo
Objeto	Fila
Identificador	Clave Primaria
Puntero u otro objeto	Clave Foránea

Esta correspondencia es muy "natural" ya que, en realidad el modelo orientado a objetos y el modelo relacional no son tan diferentes. En los programas las instancias de un objeto son accedidos a través de un puntero a su posición de memoria. Por ejemplo:

```
int a = new A;
int b =  a;
```

7.14 Thread-safe

Un *thread* es un proceso de ejecución único, es decir, un flujo individual, secuencial de control dentro de un programa. Cuando un programa es multiproceso, no estamos implicando que el programa se ejecuta en dos instancias distintas al mismo tiempo (como si al mismo tiempo de ejecutar el programa dos veces desde la línea de comandos). Más bien, estamos diciendo que la misma instancia (ejecutada sólo una vez) genera varios subprocesos que procesan este ejemplo de código. Esto significa que más de un flujo secuencial de control se ejecuta a través del mismo bloque de memoria.

Entonces, un *thread-safe* es cuando varios subprocesos ejecutan una sola instancia de un programa y por lo tanto comparten memoria, varios subprocesos podría estar tratando de leer y escribir en el mismo lugar en la memoria.

7.15 Synchronized

La palabra reservada ***synchronized*** se usa para indicar que ciertas partes del código, (habitualmente, una función miembro) están sincronizadas, es decir, que solamente un subproceso puede acceder a dicho método a la vez.

Cada método sincronizado posee una especie de llave que puede cerrar o abrir la puerta de acceso. Cuando un subproceso intenta acceder al método sincronizado mirará a ver si la llave está echada, en cuyo caso no podrá accederlo. Si método no tiene puesta la llave entonces el subproceso puede acceder a dicho código sincronizado.

7.16 JavaScript

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, se utiliza en páginas web, la sintaxis es semejante a la del lenguaje *Java* y el *lenguaje C*. *JavaScript* no es orientado a objetos, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Los navegadores interpretan el código *JavaScript* integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje *JavaScript* de una implementación del *DOM* (Modelo de Objetos de Documento, es una plataforma que proporciona un conjunto estándar de objetos a través de la cual se pueden crear documentos *HTML* y *XML*, navegar por su estructura, y modificar, añadir y borrar tanto elementos como contenidos del él mismo).

8. Herramientas

8.1 Programación Orientada a Objetos

La *POO* difiere de la programación estructurada, en que los datos y los procedimientos están separados y sin relación, y lo único que se busca es el procesamiento de datos de entrada para obtener otros de salida. La programación estructurada está en términos de procedimientos o funciones, sólo se escriben funciones que procesan datos.

La programación orientada a objetos (*POO*) es un modelo de programación que utiliza objetos, ligados mediante mensajes, para la solución de problemas. Puede considerarse como una extensión natural de la programación estructurada en un intento de potenciar los conceptos de modularidad y reutilización del código.

La programación orientada a objetos es otra forma de descomponer problemas en acciones (verbos).

Los elementos básicos de la programación orientada a objetos son: objetos, mensajes, métodos y clases.

En un programa, un ***objeto*** es la representación de un concepto, y contiene toda la información necesaria para abstraerlo: estado (datos) que describen sus atributos y comportamiento (métodos), que pueden realizarse sobre los mismos:

1. El estado está compuesto de datos, serán uno o varios atributos a los que se asignan valores concretos (datos).
2. El comportamiento está definido por los métodos o mensajes a los que responden dicho objeto, es decir, qué operaciones se pueden realizar con él.
3. La identidad es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).
4. Interfaz. Es la parte del objeto visible o accesible para el resto los objetos. También se considera el protocolo para comunicarnos con el objeto. Puede estar formado por un método o un conjunto de ellos.

El *objeto* contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

Figura 2.6 ***Representación grafica de un objeto.***

Una ***clase*** es un tipo de objeto definido por el usuario. Una clase equivale a la generalización de un tipo específico de objetos. Por ejemplo, en un molde para hacer flanes; el molde es la clase y los flanes los objetos.

Un objeto de una determinada clase se crea en el momento en que se define una variable de dicha clase. La siguiente línea declara el objeto cliente1 de la clase o tipo Cuenta.

Cuenta cliente1 = new Cuenta();

El término instancia se emplea en el sentido de que una instancia es la representación concreta y específica de una clase; por ejemplo cliente1 es una instancia de la clase Cuenta. Desde el punto este punto de vista, los términos instancia y objeto son lo mismo.

Características que contempla la "orientación a objetos":

1. ***Abstracción***: indica características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos también pueden ser abstraídos, hay una variedad de técnicas que son requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar. Es la visión de un problema que extrae la información esencial para un determinado propósito, omitiendo, los detalles menos significativos. Un objeto es una abstracción de la realidad.

2. ***Encapsulamiento***: Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema como en paquetes de información y operaciones.

3. ***Modularidad:*** Se denomina Modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos.

4. ***Principio de Ocultación***: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas. En java se consigue gracias a los modificadores de acceso. Estos pueden afectar a la clase, a los atributos y/o métodos.

5. ***Polimorfismo***: comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Consiste en utilizar un mismo símbolo para fines distintos. Respuesta distinta frente a un mensaje dependiendo de la naturaleza del objeto.

6. ***Herencia***: las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. La herencia múltiple no existe esta se simula con las interfaces.

7. Las ***Clases Abstractas*** son aquellas que se utilizan en la herencia y sólo se utilizan para que definan la interfaz (qué hacen) y/o la implementación (cómo lo hacen) de las clases heredadas. No se podrá crear ocurrencias de ellas. Se utilizan para que las características de esta clase estén disponibles para sus clases heredadas. Es una forma de generalizar un comportamiento y, por lo tanto, reutilizar código. La interfaz, es decir, a qué métodos son capaces de responder las ocurrencias de sus clases heredadas. La implementación de esa interfaz podrá estar definida en la clase abstracta o en la subclase. Si una clase abstracta sólo define el interfaz, se dice que es una clase abstracta pura (en Java se denominan interfaces). Se permite la herencia múltiple de clases interfaces.

8.2 Java

Java es lenguaje de programación además es de ser una tecnología para el desarrollo de sistemas, se compone de:

1. Java Lenguaje
2. Java Platform
3. Java Tools

Figura 2.7 ***Kit de Desarrollo de Java representación grafica de acuerdo a Java SUN.***

Java es utilizado para desarrollar software, para dispositivos móviles, *applets* que se ejecutan en el navegador, juegos, empresariales (*server-side*) y aplicaciones científicas entre otras.

La plataforma Java consta de Java Virtual Machine (*JVM*), responsable de la abstracción de hardware, y un conjunto de librerías Java cuantiosa.

Las herramientas de Java incluyen: el compilador de Java, así diversas aplicaciones de ayuda que se utilizan para el desarrollo continúo (por ejemplo, depurador).

Características de Java:

1. Orientado a Objetos.
2. Independiente de la plataforma.
3. Simple.
4. Seguro – Robusto.
5. Arquitectura-neutra.
6. Interprete –Portable.
7. Alto Rendimiento- multi-hilo.
8. Distributivo.
9. Dinámico.
10. Escalable.

Orientado a Objetos: Java está codificado utilizando principios de *POO*, facilitando la modularización de código, reutilización, capacidad de prueba y rendimiento.

Interpretada / ***Portable***: la fuente Java se cumple en *bytecode* independiente de la plataforma, lo que se interpreta (compilado en código nativo) en tiempo de ejecución. El lema de Java es "escribir una vez, ejecutar en todas partes".

Simple: java tiene una sintaxis familiar, gestión automática de memoria, manejo de excepciones, herencia simple, documentación estandarizada y un conjunto muy rico de las bibliotecas.

Seguro / ***Robusto***: debido a su apoyo a la comprobación de tipos de fuentes, manejo de excepciones, y la gestión de memoria, Java es inmune a los excesos de almacenamiento intermedio, a las filtraciones de memoria, y al acceso ilegal de datos. Además, Java viene con un controlador de seguridad que proporciona un modelo de ejecución *sand-box*.

Escalable: El diseño general de Java es escalable en términos de rendimiento, y como un entorno de desarrollo.

High-performance/Multi-hilo: con su *HotSpot Just-in-Time* compilador Java puede alcanzar (o superar) la ejecución de aplicaciones nativas. Java soporta multi-hilo desarrollo *out-of-the-box*.

Dinámica: java puede cargar componentes de la aplicación en tiempo de ejecución incluso si no sabe nada acerca de ellos. Cada clase tiene una representación en tiempo de ejecución.

Distributivo: java viene con soporte para redes, así como para invocar métodos en remotas (distribuidas) a través de objetos *RMI*.

Java Básico Sintaxis:

1. Objeto: los objetos tienen estados y comportamientos. Ejemplo: Un perro tiene estados-color, nombre, raza, así como los comportamientos-menea, ladrando, comiendo. Un objeto es una instancia de una clase.

2. Clase: una clase puede ser definida como una plantilla de impresión / azul que describen los comportamientos / estados que el objeto de su soporte tipo.

3. Métodos: un método es básicamente un comportamiento. Una clase puede contener muchos métodos. Es en métodos donde las lógicas se escriben, se manipulan los datos y todas las acciones se ejecutan.

4. Variables instantáneas: cada objeto tiene su conjunto único de variables instantáneas. Un estado objeto es creado por los valores asignados a estas variables instantáneas.

8.3 Struts II

Struts es un framework que implementa el patrón de arquitectura MVC en Java.

Frameworks con MVC

El *MVC* cumple el fin particular de cualquier *framework*, el cual es proporcionar una estructura bien definida para dar soporte a un proyecto web, quedando este último más organizado y mejor desarrollado (una estructura bien definida que da soporte a un proyecto web también ayuda a que el proyecto sea organizado y bien desarrollado).

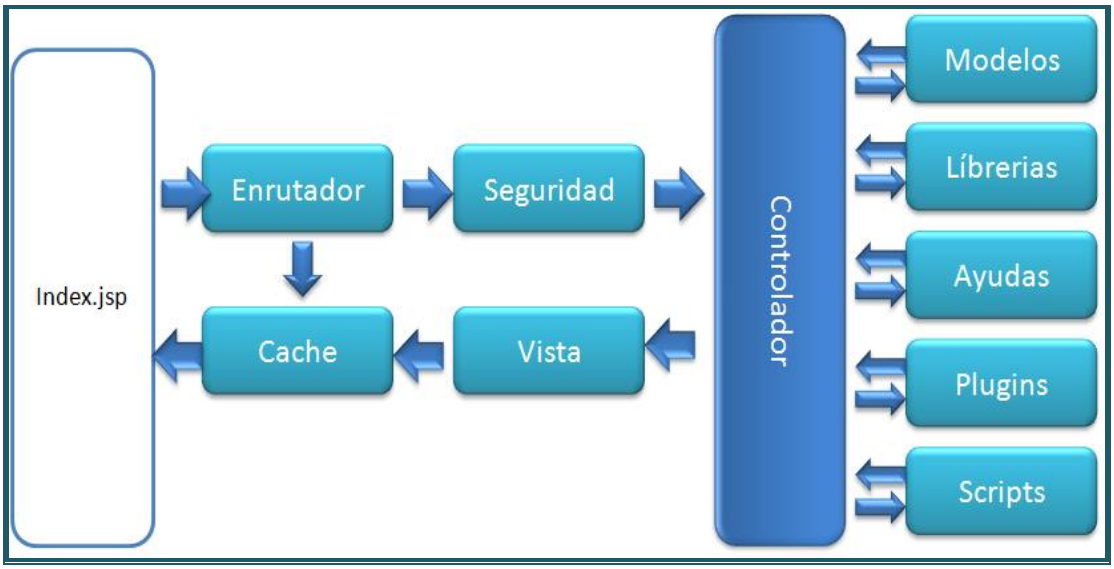


Figura 2.8 ***Diagrama de Flujo de un Framework MVC.***

Ejemplo:

1. El index.jsp nos inicializa el núcleo de nuestra aplicación.
2. El enrutador examina la petición *HTTP* y nos ayuda a determinar que se debe hacer.
3. Si existe, la cache nos devuelve nuestro archivo *HTML* sin necesidad de pasar por el sistema, ahorrándonos la carca que esto nos conlleva.
4. La seguridad, ya que antes de que se cargue el controlador se filtran los datos enviados para que estos puedan resultar fiables.
5. El controlador nos carga el *modelo, librerías*, *helpers*, *plugins* y todos los demás recursos necesarios para satisfacer nuestra petición.
6. Finalmente, cuando la Vista está renderizada, esta es enviada al navegador, entonces si la cache se encuentra habilitada, se almacena el resultado para la próxima ocasión que la *URL* sea servida.

La finalidad del *MVC* es mejorar la reusabilidad por medio del desacople entre la vista y él modelo. Sus elementos son los siguientes:

1. *El modelo es el responsable de:*
    a. Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
    b. Definir las reglas de negocio (la funcionalidad del sistema).
    c. Llevar un registro de las vistas y controladores del sistema.
    d. Si estamos ante un modelo activo, notificar a las vistas los cambios que en los datos pueda producir un agente externo.

2. *El controlador es el responsable de:*
    a. Recibir los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
    b. Contener las reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_entrega (nueva_orden_de_venta)".

3. *Las vistas son responsables de:*
    a. Recibir datos del modelo y mostrarlos muestra al usuario.
    b. Proporcionar un registro de él controlador asociado.
    c. Dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Ventajas de utilizar el MVC

1. Permite tener diferentes vistas para un mismo modelo (ejemplo: representación de un conjunto de datos como lo son una tabla o un diagrama de barras).
2. Permite construir nuevas vistas sin necesidad de modificar el modelo subyacente.
3. Proporciona un mecanismo de configuración a componentes complejos muchos más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción).

En las aplicaciones web es frecuente utilizar el *MVC*, donde la vista es la página *HTML* y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio y el controlador es el responsable de recibir los eventos entrada desde la vista.

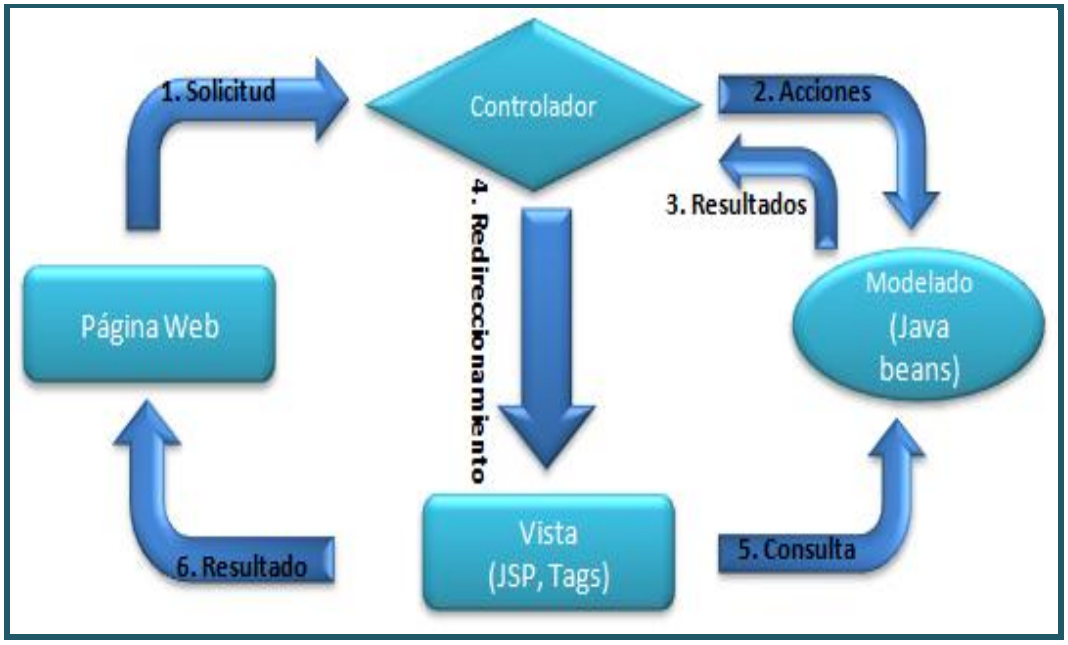


Figura 2.9 ***Diagrama Web de MVC.***

Struts provee la infraestructura básica para la implementación del patrón *MVC*, permitiendo así que los desarrolladores puedan concentrarse en la lógica de negocios.

En una aplicación el navegador genera una solicitud que es atendida por el Controlador (un *Servlet* especializado). El mismo se encarga de analizar la solicitud, seguir la configuración que se le ha programado en su *XML* y llamar al *Action* correspondiente pasándole los parámetros enviados. El *Action* instanciará y/o utilizará los objetos de negocio para concretar la tarea. Según el resultado que retorne el *Action*, el Controlador derivará la generación de interfaz a una o más JSP's, las cuales podrán consultar los objetos del Modelo a fines de realizar su tarea.

Evidentemente sirve, como todo *framework*, simplifica notablemente la implementación de una arquitectura según el patrón MVC. El mismo separa muy bien lo que es la gestión del *workflow* de la aplicación, del modelo de objetos de negocio y de la generación de la interfaz.

El controlador ya se encuentra implementado por *Struts*, aunque si fuera necesario se puede heredar y ampliar o modificar, y el *workflow* de la aplicación se puede programar desde un archivo *XML*.

Las acciones que se ejecutarán sobre el modelo de objetos de negocio se implementan basándose en clases predefinidas por el *framework* y siguiendo el patrón *Facade* (proporciona una interfaz unificada para un conjunto de interfaces de un sistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar).

Y la generación de interfaz se soporta mediante un conjunto de *Tags* predefinidos por *Struts* cuyo objetivo es evitar el uso de *Scriplets* (los trozos de código Java entre "<%" y "%>"), lo cual genera ventajas de mantenibilidad y de perfomance (pooling de Tags, caching, etc).

Logísticamente, separa claramente el desarrollo de interfaz del *workflow* y lógica de negocio permitiendo desarrollar ambas en paralelo o con personal especializado.

También es evidente que potencia la reutilización, soporte de múltiples interfaces de usuario (*Html*, *sHtml, Wml, Desktop applications,* etc.) y de múltiples idiomas, localismos, etc.

Las aplicaciones basadas en *Struts* consistirán de:

a. Código Java
b. Deployment descriptors que configuran la *framework* para el uso de nuestra aplicación.

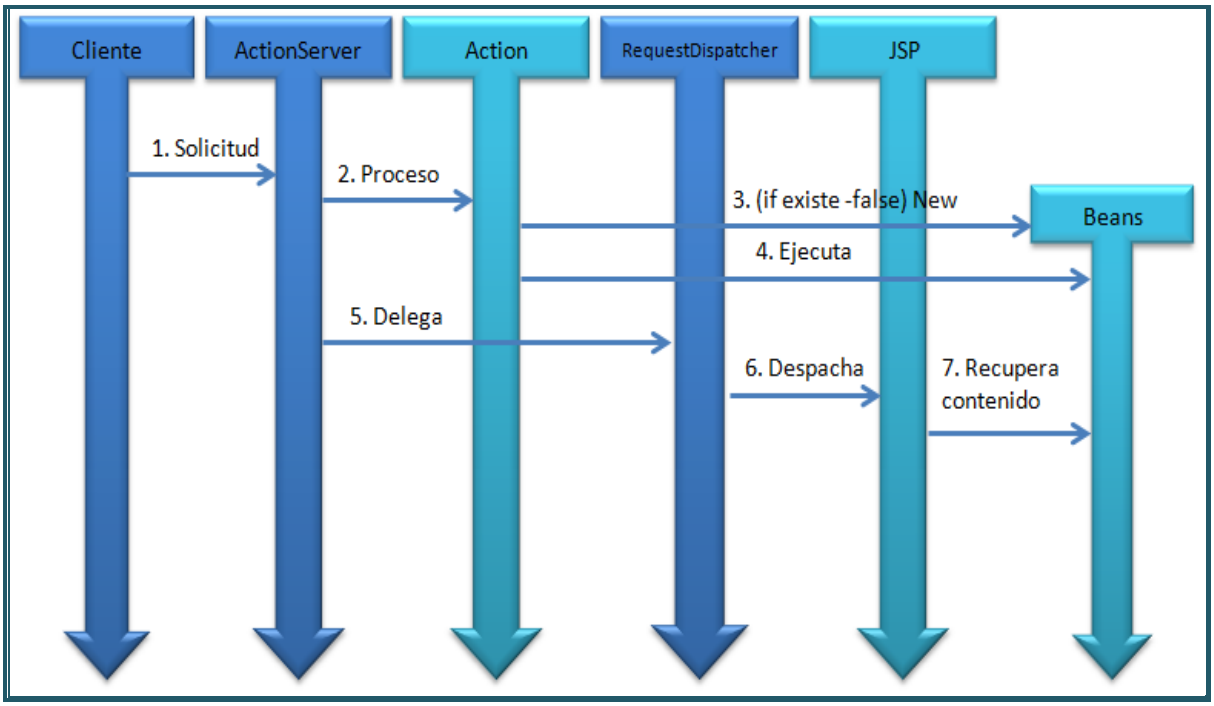


Figura 2.10 ***Control de Flujo Struts 2 de acuerdo a Apache Software Foundation.***

Struts soporta internacionalización a través de ficheros de recursos, sus librerías de etiquetas personalizadas y Java Locales.

Struts2

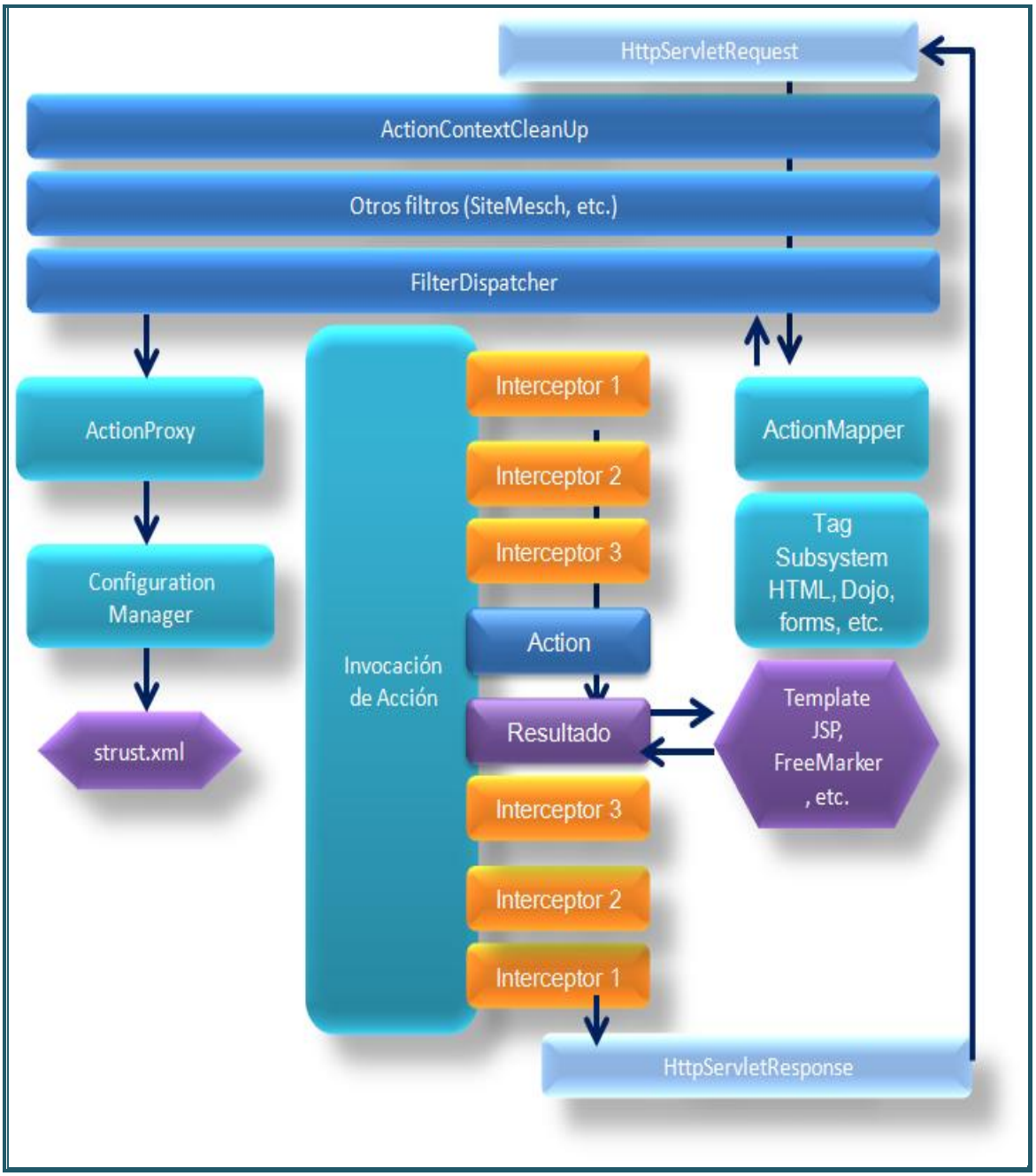


Figura 2.11 ***Arquitectura de Struts 2 de acuerdo a Apache Software Foundation.***

- Basado en el patrón *MVC*.
- *Struts 2* no se basa en *Strust 1.x* se basa en otro *framework* llamado *webwork*.
- Más simple y mucho más completo que *Strust 1.x*
- Permite reducir el tiempo de desarrollo.
- Todas las ventajas de usar un *framework*.
- Desarrollo más controlado y homogéneo.

	Struts 1	**Struts 2**
Clases Action	Se requiere extender de una clase ***Action*** abstracta. El problema común en ***Struts1*** es que se programan clases abstractas en lugar de interfaces.	Un ***Action*** puede implementar una interfaz ***Action***, además de otras interfaces que le permitan integrar más servicios. ***Struts2*** provee de una clase base ***ActionSupport*** que implementa las interfaces comúnmente usadas. Además esta interfaz no es requerida. Cualquier ***POJO*** con un método de ejecución puede ser usado como ***Action*** en ***Struts2.***
Modelo de hilos	Las ***Actions*** implementan el patrón *singleton* y deben ser ***thread-safe*** por lo que puede haber solo una instancia de una clase para manejar todas las peticiones para ese ***Action***. La estrategia *singleton* pone restricciones sobre lo que se puede hacer con los ***Actions*** de Struts1 y se requiere un cuidado extra para desarrollar. Los recursos usados para los ***Actions*** deben ser ***thread-safe*** o ***sincronized***.	Los objetos ***Action*** son instanciados por cada petición, por lo que no hay problemas de tipo con la seguridad de hilos. En la práctica, los contenedores de *servlets* generan muchas instancias, así que un objeto mas no representa problemas de rendimiento y de impacto en el ***garbage collection***.
Dependencia de los Servlets	Los ***Actions*** tienen dependencias de la ***API*** de ***Servlets*** dado que ***HttpServletRequest*** y ***HttpServletResponse*** son pasados al método *execute* cuando la ***Action*** es invocada.	Los ***Actions*** no están acoplados al contenedor. A menudo el contexto de los *servlets* está representado por simples ***Maps***, permitiendo a los ***Actions*** ser probadas de forma aislada. Las ***Actions*** de ***Struts2*** puede mantener el uso del request y el response si se requiere. De cualquier forma, otros elementos de la arquitectura reducen o eliminan la necesidad de acceder a ***HttpServetRequest*** o a ***HttpServletResponse*** directamente.
Manejo de pruebas	El principal obstáculo para los **Actions** de ***Struts1*** es que la ejecución de métodos requiere de la ***API*** de ***Servlets***. Una extensión de terceras partes, como ***Struts TestCase***, que ofrece un conjunto de	Las ***Actions*** pueden ser probadas instanciado la ***Action***, estableciendo propiedades e invocando métodos. El soporte a la ***Inyección de Dependecias*** permite hacer las pruebas de forma simple.

	simuladores para objetos de ***Struts1***.	
Recolección de datos de entrada	Es necesario el uso de un objeto ***ActionForm*** para capturar las entradas. Como los ***Actions***, todos los ***ActionForm*** deben extender de una clase. Como algunos JavaBeans no implementan de ***ActionForm*** los desarrolladores a menudo crean clases redundantes para capturar a las entradas. Los ***DynaBeans*** pueden ser usados de forma alternativa para crear clases ***ActionForm*** convencionales, pero, los desarrolladores deben *JavaBeans* existentes.	Usa las propiedades de los ***Actions*** para manejar las entradas, elimina la necesidad de un segundo objeto de entrada. Las propiedades de entrada pueden ser objetos de una amplia variedad que incluso pueden tener sus propias variables. Las propiedades de los ***Action*** pueden ser usadas por la página Web por medio de los ***TagLibs***. ***Struts2*** soporta incluso el patrón de *ActionForm*, así como objetos y acciones ***POJO***. Tipos de objetos, incluyendo negocios u objetos de dominio, pueden ser usados como objetos en entrada y salida. La característica ***ModelDriven*** simplifica las referencias por t*aglib* a objetos ***POJO***.
Lenguaje de Expresiones	Está integrado con ***JSTL***, así que usa el ***JSTL EL***. El ***EL*** tiene una traza grafica de objetos básica y una relativamente pobre colección e indexación de soporte a propiedades.	Puede usarse ***JSTL***, pero el marco de trabajo soporta además un lenguaje de expresiones más poderoso y flexible llamado "***Object Graph Notation Language" (OGNL)***.
Vinculación de objetos de la vista	Usa el mecanismo estándar de ***JSPS*** para el vincular objetos en el contexto de las páginas.	Usa una tecnología llamada "***ValueStack***" donde los *taglibs* pueden acceder a valores sin estar acoplados a la vista donde el tipo de objeto se renderea. La estrategia del "***ValueStack***" permite reutilizar vistas gracias al rango de variables que pueden tener el mismo nombre pero tipo diferente tipos de propiedades.
Conversión de tipo	Las propiedades de los ***ActionForm*** son usualmente ***Strings***. Struts1 usa comúnmente ***Commons-Beanutils*** para la conversión de tipos. Las conversiones son configurables por clase y no por instancia.	Usa ***OGNL*** para la conversión de tipos. El marco de trabajo incluye convertidores para objetos básicos y tipos primitivos.

Validación	Soporta validación manual por medio de métodos en el ***ActionForm***, o a través de una extensión de los ***Commons Validator***. Las clases pueden tener diferentes contextos de validación para la misma clase, pero no pueden encadenar validaciones en sub objetos.	Soporta validación manual por medio de métodos y por el marco de trabajo ***XWork Validation***. El ***XWork*** soporta encadenamiento de validaciones en las subpropiedades usando las validaciones definidas para las propiedades de la clase y el contexto de validación.
Control de la ejecución del Action	Soporta ***Request Porcessors*** (Ciclos de vida) separados por cada módulo, pero todas las ***Actions*** en el módulo deben compartir el mismo ciclo de vida.	Soporta ciclos de vida diferentes, uno por ***Action*** por medio de la pila de interceptores. Pilas configurables pueden ser creadas y usadas con diferentes ***Actions*** si es necesario.

En esta tabla se observa solo algunas de las mejoras en *Struts2*, sin embargo se pondría especial énfasis en las ventajas que ofrecen los interceptores en *Struts2*, ya que permiten aislar gran parte de operaciones de control que son comunes en todas las aplicaciones Web, entre ellas una muy importante, el manejo de la sesión y la restricción de acceso a usuarios con permisos. Aislando estas operaciones es más fácil reutilizar el código en otras aplicaciones.

Además los interceptores que vienen por default en el marco de trabajo resuelven y aíslan una gran cantidad trabajo, como carga de archivos, validaciones, internacionalización y más. Solo tendremos que usar lo que los desarrolladores de *Struts2* ponen a nuestra disposición.

Ventajas

1. Formularios *POJO* (Plain Old Java Objects), ya no se utilizan los *ActionForm*.
2. Acciones *POJO*, no hace falta extender de *Action*.
3. Mejoras en los *tags*.
4. Archivo de configuración opcional y con posibilidad de dividirlo en paquetes.
5. Soporte para *AJAX*.
6. Integración de herramientas *debbuging* y *profiling*.

Desventajas

1. La documentación no está muy bien organizada.
2. No hay *feedback* para las propiedades que no se han las expresiones *OGNL* no válidas.

8.4 Hibernate

Hibernate es una herramienta que realiza el *mapping* entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las bases de datos en entornos *Java*. El término utilizado es *ORM* (object/relational mapping) y consiste en la técnica de realizar la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa.

Hibernate no solo realiza esta transformación sino que nos proporciona capacidades para la obtención y almacenamiento de datos de la base de datos que nos reducen el tiempo de desarrollo.

Hibernate es una capa de persistencia objeto/relacional y un generador de sentencias sql permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada podremos generar *BD* en cualquiera de los entornos soportados: *Oracle, DB2, MySql*, etc. Y lo más importante de todo, es open source, lo que supone, entre otras cosas, que no tenemos que pagar nada por adquirirlo.

Uno de los posibles procesos de desarrollo consiste en, una vez tengamos el diseño de datos realizado, mapear este a ficheros *XML* siguiendo la *DTD* de mapeo de *Hibernate*. Desde estos podremos generar el código de nuestros objetos persistentes en clases Java y también crear BD independientemente del entorno escogido.

Figura 2.33 ***Estructura de Hibernate de acuerdo a Hibernate.***

Hibernate utiliza la *BD* y la configuración de los datos para proporcionar servicios y objetos persistentes a la aplicación que se encuentre justo por arriba de él.

Conceptos básicos de Hibernate

Hibernate funciona asociando a cada tabla de la base de datos un Plain Old Java Object (*POJO*, a veces llamado Plain Ordinary Java Object). Un *POJO* es similar a una *Java Bean*, con propiedades accesibles mediante métodos *setter* y *getter*, como por ejemplo:

```
package net.sf.hibernate.examples.quickstart;

public class Gato {

    private String id;
    private String nombre;
    private String color;

    public Gato() {
    }

    public String getId() {
        return id;
    }

    private void setId(String id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
}
```

Para asociar el *POJO* a su tabla correspondiente en la base de datos, *es necesario agregar las anotaciones correspondientes, para relacionar las* columnas en la base de datos, asociación de tipos de datos, referencias, relaciones x a x con otras tablas, etc. Para el ejemplo anterior esto quedaría como:

```
package net.sf.hibernate.examples.quickstart;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table (name="Gato")
public class Gato {
```

```
    private String id;
    private String nombre;
    private String color;

   @Id
   @Column(name="id")
    public String getId() {
        return id;
    }

    private void setId(String id) {
        this.id = id;
    }

    @Column (name="NOMBRE" , nullable=false)
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    @Column (name="COLOR" , nullable=false)
    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
}
```

De esta forma en la aplicación podemos usar el siguiente código para la comunicación con la base de datos:

```
sessionFactory = new
Configuration().configure().buildSessionFactory();
Session session = sessionFactory.openSession();

Transaction tx= session.beginTransaction();

Gato princess = new Gato();
princess.setNombre("Princess");
princess.setColor('Red');

session.save(princess);
tx.commit();

session.close();
```

Además tiene la ventaja de que es totalmente transparente el uso de la base de datos pudiendo cambiar de base de datos sin necesidad de cambiar una línea de código de la aplicación, simplemente cambiando los ficheros de configuración de *Hibernate*.

Configuración de Hibernate

Para la configuración de la base de datos mediante hibernate se puede usar un fichero *hibernate.properties* o *hibernate.cfg.xml* que debe estar en el *path* de la aplicación:

hibernate.properties:

```
## MySQL

hibernate.dialect net.sf.hibernate.dialect.MySQLDialect
hibernate.connection.driver_class org.gjt.mm.mysql.Driver
hibernate.connection.driver_class com.mysql.jdbc.Driver
hibernate.connection.url jdbc:mysql:///test
hibernate.connection.username cesar
hibernate.connection.password
```

hibernate.cfg.xml - Conexión mediante Datasource:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration
PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-
3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.datasource">
        java:comp/env/jdbc/shop</property>

    <property name="dialect">net.sf.hibernate.dialect.MySQLDialect
        </property>
    <property name="use_outer_join">true</property>
    <property name="transaction.factory_class">

net.sf.hibernate.transaction.JDBCTransactionFactory</property>

    <property name="show_sql">true</property>

    <!-- Mapping files -->
    <mapping class="com.shop.Category" />
    <mapping class="com.shop.Product" />
  </session-factory>
</hibernate-configuration>
```

hibernate.cfg.xml - Conexión Directa:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration
PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-
3.0.dtd">

<hibernate-configuration>
<session-factory>
<property name="connection.driver_class">
com.mysql.jdbc.Driver</property>
<property name="connection.url">
jdbc:mysql://localhost/test</property>
<property name="connection.username">cesar</property>
<property name="connection.password"></property>

<property
name="dialect">net.sf.hibernate.dialect.MySQLDialect</property>
<property name="use_outer_join">true</property>
<property
name="transaction.factory_class">net.sf.hibernate.transaction.JDBC
TransactionFactory</property>
<property name="show_sql">true</property>

<!-- Mapping files -->
    <mapping class="com.shop.Category" />
    <mapping class="com.shop.Product" />
  </session-factory>
</hibernate-configuration>
```

Desde la aplicación también se puede especificar el fichero a usar:

```
SessionFactory sf = new Configuration()
    .configure("catdb.cfg.xml")
    .buildSessionFactory();
```

En estos ficheros se indican los parámetros de conexión de la base de datos como la base de datos a la que conectar, usuario y password etc.

Un parámetro interesante es el Dialecto de *Hibernate*. En este parámetro se indica el nombre de la clase que se encargará de comunicarse con la base de datos en el *SQL* que entienda la base de datos. Este parámetro ha de ser siempre especificado. El valor ha de ser una subclase que herede de *net.sf.hibernate.dialect.Dialect*.

Hibernate nos proporciona los siguientes dialectos:

RDBMS	Dialect
DB2	net.sf.hibernate.dialect.DB2Dialect
MySQL	net.sf.hibernate.dialect.MySQLDialect
SAP DB	net.sf.hibernate.dialect.SAPDBDialect
Oracle (any version)	net.sf.hibernate.dialect.OracleDialect
Oracle 9	net.sf.hibernate.dialect.Oracle9Dialect
Sybase	net.sf.hibernate.dialect.SybaseDialect
Sybase Anywhere	net.sf.hibernate.dialect.SybaseAnywhereDialect
Progress	net.sf.hibernate.dialect.ProgressDialect
Mckoi SQL	net.sf.hibernate.dialect.MckoiDialect
Interbase	net.sf.hibernate.dialect.InterbaseDialect
Pointbase	net.sf.hibernate.dialect.PointbaseDialect
PostgreSQL	net.sf.hibernate.dialect.PostgreSQLDialect
HypersonicSQL	net.sf.hibernate.dialect.HSQLDialect
Microsoft SQL Server	net.sf.hibernate.dialect.SQLServerDialect
Ingres	net.sf.hibernate.dialect.IngresDialect
Informix	net.sf.hibernate.dialect.InformixDialect
FrontBase	net.sf.hibernate.dialect.FrontbaseDialect

Aquí se observa la gran importancia del fichero de configuración, pues es aquí donde se especifica que base de datos será utilizada, por lo que si se cambia de base de datos bastaría con cambiar este fichero de configuración, manteniendo la aplicación intacta.

Hibernate Query Languaje HQL

Hibernate proporciona además un lenguaje con el que realizar consultas a la base de datos.

Este lenguaje es similar a *SQL* y es utilizado para obtener objetos de la base de datos según las condiciones especificadas en el *HQL*.

El uso de *HQL* permite usar un lenguaje intermedio que según la base de datos que se use y el dialecto que se especifique será traducido al *SQL* dependiente de cada base de datos de forma automática y transparente.

Así una forma de recuperar datos de la base de datos con Hibernate sería:

Hibernate	JDBC

`Session session = sessionFactory.openSession();`
`List cats = null;`
`try {`
	`Driver d = (Driver) Class.forName("com.mysql.jdbc.Driver").newInstance();`

```
categories =
session.find("from Cat");

Iterator i =
categories.iterator();
while (i.hasNext() == true) {
Cat cat = (Cat)i.next();
...
...
}
} finally {
session.close();
}
```

```
DriverManager.registerDriver(d);


try {
Connection con =
DriverManager.getConnection(
"jdbc:mysql://yamcha/test",
"cesar",
"");

Statement stmt =
con.createStatement();

String select =
"SELECT * from cat";

ResultSet res =
stmt.executeQuery(select);



while (res.next() == true) {
String catID = res.getString("id");
String catName =
res.getString("name");

Cat cat = new Cat(catID,catName);
(.......)

list.add(cat);

}
stmt.close();
con.commit();
con.close();

} catch (Throwable ex) {
System.out.println(" Error
visualizando datos ");
}
```

De esta forma:

HQL	**SQL**
`from Cat`	`select * from cat`

Como se puede observar se simplifica considerablemente el código, así como se desacopla el uso de la base de datos de la lógica de aplicación.

Criteria

Hibernate proporciona formas alternativas de manipulación de objetos y, a su vez dispone de datos en tablas *RDBMS* (Relational Database Management System). Uno de los métodos es *Criteria*, es una *API* que permite construir un objeto de consulta a través de los criterios de programación, donde se pueden aplicar reglas de filtrado y las condiciones lógicas.

La interfaz *Session Hibernate* proporciona un método *createCriteria*() que se puede utilizar para crear un objeto *Criteria* que devuelve instancias de la clase del objeto persistencia cuando la aplicación ejecuta una consulta de criterios.

A continuación se muestra un ejemplo de una consulta de criterios que devuelve todos los objetos que correspondan a la clase *Employeer*:

```
Criteria cr = session.createCriteria(Employeer.class);
List results = cr.list();
```

La clase POJO *Employeer* se define como:

```
public class Employeer {
   private int id;
   private String firstName;
   private String lastName;
   private int salary;

   public Employee() {}
   public Employee(String fname, String lname, int salary) {
      this.firstName = fname;
      this.lastName = lname;
      this.salary = salary;
   }
   public int getId() {
      return id;
   }
   public void setId( int id ) {
      this.id = id;
   }
   public String getFirstName() {
      return firstName;
   }
   public void setFirstName( String first_name ) {
      this.firstName = first_name;
   }
   public String getLastName() {
      return lastName;
   }
   public void setLastName( String last_name ) {
      this.lastName = last_name;
```

```
    }
    public int getSalary() {
       return salary;
    }
    public void setSalary( int salary ) {
       this.salary = salary;
    }
}
```

8.5 MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario.

Características:

Inicialmente, *MySQL* carecía de elementos como integridad referencial y transacciones considerados esenciales en las bases de datos relacionales. Atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad.

Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje *SQL*. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferente velocidad de operación, soporte físico, capacidad, distribución geográfica, transacciones.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

MySQL es un sistema de administración de bases de datos. Esta puede ser desde una simple lista de compras a una galería de pinturas o el extenso volumen de información en una red corporativa.

Para agregar, acceder y procesar datos guardados en un computador, se necesita un administrador como *MySQL Server*. Dado que las computadoras manejan grandes cantidades de información, los administradores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones.

MySQL es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

Además *MySQL* es un software de fuente abierta. *MySQL* usa el *GPL* (*GNU* General Public License) para definir qué puede hacer y qué no puede hacer con el software en diferentes situaciones. Si el

acoplamiento de *GPL* es complicado o se requiere introducir código *MySQL* en aplicaciones comerciales, es posible comprar una versión comercial licenciada.

Características distintivas

Las siguientes características son implementadas únicamente por *MySQL*:

- Permite escoger entre múltiples motores de almacenamiento para cada tabla. En *MySQL* 5.0 éstos debían añadirse en tiempo de compilación, a partir de *MySQL* 5.1 se pueden añadir dinámicamente en tiempo de ejecución; los hay nativos como *MyISAM, Falcon, Merge, InnoDB, BDB, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole*.
- Desarrollos por socios como: solidDB, NitroEDB, ScaleDB, TokuDB, Infobright (antes Brighthouse), *Kickfire*, *XtraDB*, *IBM DB2*). *InnoDB* Estuvo desarrollado así pero ahora pertenece también a Oracle.
- Desarrollados por la comunidad como *memcache, httpd, PBXT y Revision*.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

Tipos de compilación del servidor

Hay tres tipos de compilación del servidor *MySQL*:

- Estándar: Los binarios estándar de *MySQL* son los recomendados para la mayoría de los usuarios, e incluyen el motor de almacenamiento *InnoDB*.
- Max (No se trata de MaxDB, que es una cooperación con *SAP*): Los binarios incluyen características adicionales que no han sido lo bastante probadas o que normalmente no son necesarias.
- MySQL-Debug: Son binarios que han sido compilados con información de depuración extra. No debe ser usada en sistemas en producción porque el código de depuración puede reducir el rendimiento.

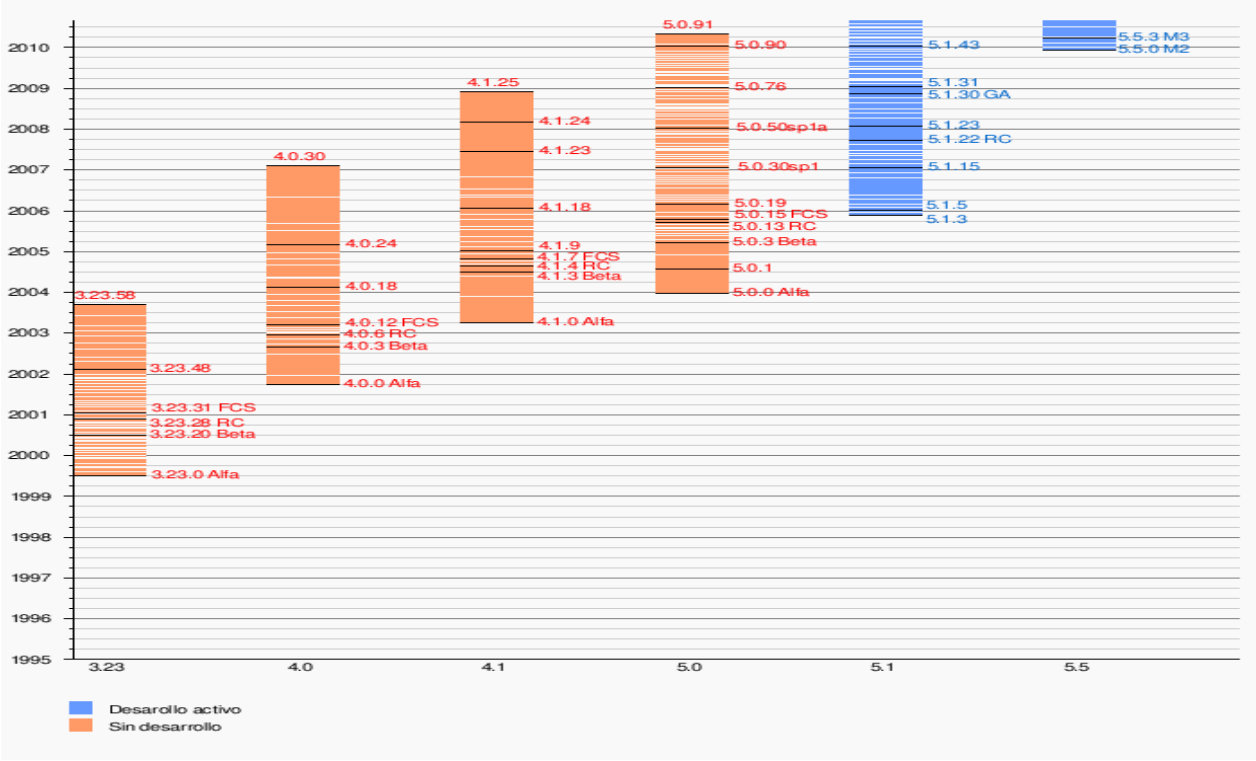


Figura 2.34 ***Versiones de MySql recurso de Wikipedia.***

Capítulo III – Contexto del SCCEM

1. ANTECEDENTES

En México debido a diversos factores han cambiado las enfermedades a las que nos enfrentamos hoy día siendo algunas de ellas muy costosas, para el paciente como para el Gobierno derivándose él empobrecimiento y la mortandad de la población, por tal motivo el Gobierno se ha dado a la tarea de mejorar estas condiciones como son; implementación de Programas de Calidad para garantizar el mejoramiento de los servicios de salud, entre otras innovaciones.

Es así que en el Plan Nacional de Salud se definen compromisos que determinan los avances, formas de atención, evaluación del cumplimiento de los objetivos, acciones y metas establecidas en materia de: equidad de género, salud reproductiva, planificación familiar, cáncer cérvico uterino, cáncer mamario, salud materna y perinatal, prevención y atención de la violencia familiar o sexual contra las mujeres.

El Centro Nacional de Equidad de Género y Salud Reproductiva (**CNEGySR**) se define como un órgano desconcentrado de la Secretaría de Salud, que implementa programas, como el de la salud sexual y reproductiva, con la misión de contribuir a incorporar la perspectiva de género en salud y a mejorar la salud sexual y reproductiva de la población a través de programas y acciones sustentados en evidencias científicas, mejores prácticas y las necesidades de salud de la población, con un sentido humanitario y compromiso social, en un marco de participación social y con el pleno respeto a las garantías individuales de nuestra Constitución.

En 1996 se reubica el Programa de Cáncer en la Mujer de la Dirección General de Medicina Preventiva a la Dirección General de Salud Reproductiva, para formar parte de los componentes de Salud Reproductiva y se reorienta y renueva el programa incluyendo componentes de normatividad, estrategias programáticas, capacitación, control de calidad, información y evaluación, investigación, comunicación educativa y social y participación comunitaria. Posteriormente en el Diario Oficial se publicó la creación del Comité Nacional para la Prevención y Control del Cáncer Cérvico Uterino y Mamario, con el objeto de contribuir a la disminución de las tasas de mortalidad por dichas neoplasias mediante la intensificación de acciones preventivas y de control de factores de riesgo, así como el desarrollo de estrategias para su detección oportuna. Se elaboró un nuevo Programa de Prevención y Control de Cáncer Cérvico Uterino para el periodo 1998-2000, incluyendo avances científicos y tecnológicos que fortalecieron la estructura y los recursos disponibles para su operación. Después, el Programa de Cáncer de Mama se separó del Programa de Prevención y Control de Cáncer Cérvico Uterino, debido a que los factores de riesgo, estilos de vida y los especialistas vinculados a cada uno de estos cánceres son muy distintos así como el incremento paulatino del cáncer de mama como una enfermedad emergente vinculada a la modernidad.

La Asociación de Lucha contra el Cáncer apoya al Programa Nacional de Prevención y Control de Cáncer integrado a la Dirección General de Medicina Preventiva conformada por una Subdirección y tres Jefaturas de Departamento, con tres componentes: a) Prevención de Cáncer Cérvico Uterino; b) Prevención de Cáncer en órganos digestivos y peritoneo y; c) Prevención de Cáncer de aparato genitourinario y tejido linfático de los órganos hematopoyéticos, dando mayor énfasis a la prevención del cáncer cérvico uterino y de mama.

El **CNEGySR** tiene las siguientes facultades:

1. Proponer y conducir la elaboración de normas oficiales mexicanas, guías técnicas en materia de salud reproductiva; salud materna y perinatal, equidad de género y violencia intrafamiliar, en el ámbito de su competencia.
2. Coordinar, supervisar, asesorar y evaluar la calidad de la prestación de los servicios de salud en materia de su competencia, así como proporcionar asistencia técnica al personal encargado de operar los programas a su cargo en las entidades federativas.
3. Participar en la definición, instrumentación, supervisión y evaluación de las estrategias y contenidos técnicos de los materiales de comunicación social en el ámbito de su competencia, así como de los materiales didácticos y metodologías que se utilizan para la capacitación, actualización y desarrollo humano del personal que opera los programas a su cargo, en coordinación con las unidades administrativas competentes.
4. Proponer los lineamientos y procedimientos técnicos para la organización, programación, como establecer políticas y mecanismos que promuevan la participación de los sectores público, social y privado en los programas de acción en el ámbito de su competencia.
5. Promover entre las unidades administrativas de la Secretaría la instrumentación de acciones que permitan incorporar la perspectiva de género en todas las actividades de las instituciones públicas del Sistema Nacional de Salud, incluidas la planeación, programación, presupuesto y prestación integral de servicios de salud; para promover el desarrollo de los programas de acción en materia de equidad de género, violencia contra las mujeres y salud reproductiva.
6. Promover, coordinar y, en su caso, efectuar investigación operativa y desarrollo tecnológico en relación con los temas del ámbito de su competencia, en coordinación con las demás unidades administrativas competentes.
7. Promover mecanismos para fomentar la participación de la sociedad civil y, en lo general de la comunidad, así como de los sectores público y privado en las acciones materia de su competencia;
8. Propiciar la celebración de convenios y acuerdos de coordinación con los gobiernos de las entidades federativas y municipios, para impulsar su apoyo y participación en el desarrollo de las acciones de los programas a su cargo;
9. Promover la cooperación y la coordinación técnica de las instituciones y organizaciones nacionales con agencias e instituciones regionales e internacionales

en relación con las actividades del ámbito de su competencia, con la colaboración de las demás unidades administrativas competentes;

10. Proponer a las instituciones del Sistema Nacional de Salud la adopción o modificación de sus políticas, lineamientos y estrategias con el propósito de lograr la equidad de género en las acciones en materia de salud, así como de enfrentar la violencia familiar, sexual y contra las mujeres como un problema de salud pública y a promover el respeto a los derechos sexuales y reproductivos.
11. Supervisar y evaluar la información generada por los sistemas institucionales de información y de investigación en salud, relacionada con las actividades de los programas a su cargo, en coordinación con las unidades administrativas competentes.
12. Certificar el desempeño de las localidades, jurisdicciones sanitarias, entidades federativas, regiones, comunidades o establecimientos, según sea el caso, en relación con los programas del ámbito de su competencia.

Por lo tanto el **CNEGySR** llega a proponer, evaluar lineamientos de programación, así como los indicadores necesarios para medir el uso adecuado de los equipos e insumos estratégicos, además de proponer temas o proyectos de investigación y desarrollo tecnológico que son proporcionados a las diferentes entidades federativas para la generación de información estratégica para la toma de decisiones, en los Programas de Acción de Salud Materna y Perinatal, Cáncer Cérvico Uterino, Mamario, Planificación Familiar y Anticoncepción Salud Sexual y Reproductiva de los Adolescentes, Peri y Posmenopausia y Violencia Familiar y de Género. Establece estrategias para promover modificaciones y/o adopciones de políticas, normas, lineamientos que faciliten el llevar acabó la participación en actividades de supervisión integral, capacitación y adiestramiento de los recursos humanos en formación y en servicio, responsables de la prestación de servicios de salud reproductiva.

La Dirección de Cáncer de la Mujer tiene como objetivo dirigir las actividades para la prestación de servicios de calidad de prevención y control de cáncer cérvico uterino y de mama, priorizando las acciones en los grupos de mayor riesgo y mayor rezago en el país, y así propiciar la equidad en salud y el mejoramiento de la salud de la mujer. Y está a cargo de coordinar la actualización de la Norma Oficial Mexicana NOM-041 SSA 2001, para la Detección, Diagnóstico, Tratamiento, Control y Vigilancia Epidemiológica del Cáncer de Mama, y de la NOM-014-SSA-2-1994, para la Detección, Prevención, Diagnóstico, Tratamiento, Control y Vigilancia Epidemiológica del Cáncer Cérvico Uterino.

Para llevar a cabo nuevos programas se requiere ejecutar proyectos de investigación que documenten la situación actual de salud y su acceso a los servicios de salud en colaboración con los organismos públicos y privados de salud.

1. Definir la programación e indicadores para la estimación de los materiales, equipos y recursos humanos necesarios para las actividades de los programas.

2. Coordinar la ejecución de estrategias gerenciales y operativas que difundan las normas oficiales mexicanas en materia de cáncer en la Mujer para mejorar la calidad de la atención e incrementar la cobertura de los servicios en las entidades federativas.
3. Coordinar la supervisión y asesoría técnica en las entidades federativas para monitorear y dar seguimiento a las políticas establecidas en los programa de acción de cáncer cérvico uterino y de mama.
4. Establecer estrategias de información, educación y comunicación, para incrementar el uso de los servicios de los Programas de cáncer cérvico uterino y de mama.
5. Coordinar la actualización de los manuales e instructivos para la prevención, diagnóstico, tratamiento, control y vigilancia epidemiológica del cáncer cérvico uterino y de mama.
6. Consolidar los programas de capacitación y desarrollo humano del personal adscrito a la Dirección de Área para una actualización permanente en los programas de cáncer cérvico uterino y de mama.
7. Establecer las estrategias de capacitación técnica y humanística del personal de salud en las entidades federativas, para consolidar la descentralización de los programas de acción de cáncer cérvico uterino y de mama.
8. Definir las acciones para la evaluación cuantitativa y cualitativa de los programas de acción de cáncer cérvico uterino y de mama.
9. Apoyar el cumplimiento de los acuerdos y compromisos con las autoridades estatales de salud, establecidos durante las visitas de supervisión.

Del mismo modo la Subdirección de Cáncer de Mama tiene como objetivo colaborar en la determinación de los criterios normativos, métodos y procedimientos para la detección, diagnóstico, tratamiento y seguimiento del cáncer de mama, así como verificar su cumplimiento, con el propósito de mejorar la atención médica de la mujer para disminuir al máximo la mortalidad por cáncer de mama. Así también verificar la implementación y observancia de la Norma Oficial Mexicana NOM-041-SSA2-2009, para la prevención, diagnóstico, tratamiento, control y vigilancia epidemiológica del cáncer de mama.

La subdirección está encargada de coordinar la capacitación continua, elaborar y establecer lineamientos de programación e indicadores para la estimación de materiales, equipos y recursos humanos, coordinar con las organizaciones de la sociedad civil la promoción y difusión de información sobre cáncer de mama, además de implementar y difundir el Sistema de Información de Cáncer de la Mujer (SICAM-PROMAMA) para registrar las acciones generadas en el Programa de Acción de Cáncer de Mama, en las entidades federativas y poder realizar su seguimiento y elaborar reportes de cumplimiento de metas, calidad de los procesos, oportunidad de diagnóstico y seguimiento de pacientes con sospecha de cáncer de mama, que sirvan para la adecuada toma de decisiones hacia el cumplimiento de las metas del programa.

Como el SICAM-PROMAMA se encuentra en las primeras etapas de implementación se requiere de actualizaciones para optimizar su funcionamiento y capacitar al personal involucrado de las

entidades federativas en el manejo y aplicación del sistema de información para el monitoreo de los indicadores del programa de Cáncer de Mama.

La evaluación para conocer los avances será a través de los indicadores del Programa de Acción para la Prevención y Control del Cáncer de Mama y así poder mejorar el programa de Cáncer de Mama en forma continua.

La subdirección está encargada de mantener actualizados los manuales e instructivos para la detección, diagnóstico, tratamiento, control y vigilancia epidemiológica del cáncer de mama para incrementar la cobertura de manera eficiente, segura y eficaz. Uno de éstos es el de ***Control de Calidad de Mastografía*** el cual pretende garantizar la calidad de la imagen mamográfica, evitando diagnósticos erróneos asegurando la integridad del paciente, para lo cual se requiere equipo especializado con tecnología de punta, personal médico y técnico altamente capacitado, como garantizar una radiación mínima tanto para el paciente como para el personal a cargo, además de considerar que los establecimientos radiológicos cumpla con las especificaciones mínimas necesarias de acuerdo a las NOM-229-SSA1-2002, NOM-041-SSA2-2011 [SSA 11], así como estándares internacionales. Y ya que esté manual es una las innovaciones del programa de calidad, se requiere elaborar acciones para su implementación en el sector salud, siendo el primero de su tipo en México en donde la tendencia de este programa es principalmente obtener información fidedigna, en tiempo real acerca del funcionamiento de los equipos para tomar decisiones estratégicas acerca de los recursos, pero sobre todo garantizar la calidad de los servicios médicos. **Esto sugiere que no se cuenta con un registro certero de tamizaje en México.**

La implementación de este programa se enfatiza en las Unidades de Especialidades Médicas de Detección y Diagnóstico de Cáncer de Mama (UNEME-DEDICAM), en las que se ofrece a las usuarias un servicio profesional, moderno y oportuno desde una perspectiva humanística, procurando su bienestar psicológico y social, con personal médico especializado y tecnología de vanguardia, para agilizar el desempeño del personal, mejorar sensiblemente la calidad de imagen y la precisión diagnóstica.

El equipo instalado tiene la capacidad necesaria para detectar lesiones milimétricas, lo cual facilita el tratamiento haciéndolo menos invasivo y agresivo para la paciente, permitiéndole un pronóstico preciso y calidad de vida mucho más prolongada.

Para él, control de calidad de la prueba de tamizaje y los mecanismos para la referencia y tratamiento de pacientes con resultados anormales se requiere un sistema de información para la organización del tamizaje con base poblacional, el seguimiento de pacientes con resultados anormales y el monitoreo y evaluación del programa, tanto pacientes como personal de salud frecuentemente no se apegan a las recomendaciones de los programas de tamizaje, por lo que es necesario un sistema de seguimiento y evaluación que considere de forma integral todas las posibles fallas y debilidades para incorporar los sistemas de control de calidad que permitan la mayor eficacia de los recursos, así como resultados benéficos para la población.

En relación con el cáncer de mama existe suficiente evidencia científica que confirma que en países desarrollados, un programa de tamizaje organizado y realizado de manera óptima tiene la capacidad de reducir hasta 35% las tasas de mortalidad y la carga de la enfermedad en la población. Para lograr el impacto mencionado son esenciales servicios de alta calidad, que pueden ser alcanzados si se consideran principios básicos ya establecidos. Desde un punto de vista ético, estos principios deben tener el mismo valor que cuando se aplican a servicios de diagnóstico clínico.

Por las características del cáncer de mama, a las mayores de 40 años de edad con o sin síntomas mamarios se les realiza una mastografía de tamizaje, es decir, aunque no presenten molestias, o una mastografía diagnóstica si llega con molestias o tumores palpables. A las mujeres que tienen diagnóstico confirmatorio o el resultado de una mastografía que es sospechoso, se le realizan estudios de ultrasonografía o análisis a través de una Biopsia.

2. PROBLEMÁTICA

El **CNEGySR** con las facultades que cuenta y siendo parte del Plan Nacional de Salud, y teniendo como objetivo colaborar, desarrollar, integrar e implementar los Programas de Calidad y del Programa de Acción para la Prevención y Control del Cáncer de Mama para definir criterios normativos, métodos y procedimientos para la detección, diagnóstico, tratamiento y seguimiento del cáncer de mama, elaboró el **Manual de Control de Calidad en Mastografía** con mesas de trabajo, personal experto, de acuerdo a la NOM-229-SSA1-2002, la NOM-041-SSA2-2011 [SSA 11], y estándares internacionales, buscando la implementación en las UNEMES a nivel federal para contribuir a las buenas prácticas hospitalarias, en el área de la imagen de mama.

En este manual se documentaron las pruebas para los equipos radiológicos de tipo analógico y digital. Las pruebas se diseñaron para el registro manual de los protocolos, por lo tanto en la implementación se presentó la resistencia al cambio, deficiencia e insuficiencia de personal, la capacitación del personal, limitaciones en la operación y líneas de acción, por lo tanto se requiere generar un proceso de recolección de información robusto para facilitar la toma de decisiones estratégicas. Por esta razón se vuelve prioritario que esta información esté al alcance del personal experto de manera más eficaz y eficiente para que se puedan tomar las decisiones necesarias y brindar servicios de salud de calidad.

La Asociación Mexicana de Lucha contra el Cáncer apoya al Programa Nacional de Prevención y Control de Cáncer, brindó un donativo al **CNEGySR** buscando la implementación del **Manual de Control de Calidad en Mastografía,** ya que cumple con la prevención y evaluación de diferentes indicadores en el control de calidad.

Constantemente se están implementando y definiendo procesos de control de calidad, por lo cual se buscó una solución de software que respaldara de forma eficiente las actividades involucradas en la obtención de la información oportuna, para de esta manera tomar decisiones estratégicas para la optimización de los recursos.

Con lo anterior se visualiza la oportunidad de sistematizar el registro de las pruebas de los equipos, eliminando el uso del papel, homogeneizando la forma de registro por parte del personal técnico y haciendo más sencilla la consulta de los resultados por parte del personal experto encargado de la supervisión. Se espera un desarrollo de software fácil de usar para que la capacitación del personal sea transparente y rápida.

Dentro de las ventajas que se pretenden tener se encuentran las siguientes:

1. Recabar información de manera oportuna.
2. Consultar la información para la toma de decisiones estratégicas acerca de los recursos.
3. Garantizar la calidad de la imagen mamográfica.
4. Validar que el equipo funcione correctamente y con ello garantizar una radiación mínima para el personal.

5. Evitar diagnósticos erróneos.
6. Tener registros históricos del funcionamiento de los equipos.
7. Llevar el control en la evaluación de los equipos.

3. PLANTEAMIENTO DE LA SOLUCIÓN

De acuerdo con el **Manual de Control de Calidad en Mastografía** para mastógrafos analógicos y digitales se planteó básicamente hacer el registro de las pruebas de acuerdo a las definiciones establecidas, considerando que se tendrán que implementar en las UNEMES a nivel federal, lo que también ayudaría a la sustentación del proceso de registro.

Por lo tanto se planteó desarrollar un sistema a través de Internet para lograr una mayor eficacia en el registro de pruebas, y que el personal del **CNEGySR** pudiera consultarlo de manera oportuna.

Se propuso que además de los registros de las pruebas se contara con la gestión de la programación de las mismas, así como llevar un control de dichas pruebas de acuerdo al número de equipos para cada UNEME por entidad federativa.

La solución del software que se proyectó consistió en un conjunto de actividades organizadas para conseguir el mismo objetivo que propone el **Manual de Control de Calidad en Mastografía**. Una primera etapa consistió en realizar el análisis para determinar el alcance del proyecto, así como la síntesis del conocimiento adquirido acerca del proceso, esto nos permitió diseñar el flujo de datos, incluidos el número de módulos, roles y jerarquías de la información.

Los módulos planteados son:

1. **Control de acceso.**
2. **Administración de usuarios.**
3. **Catálogos.**
4. **Administración de las pruebas**
5. **Generación Física de las Pruebas.**
6. **Control de pruebas.**
7. **Captura de Pruebas.**
8. **Monitorear Pruebas.**

De esta forma se planea tener el control de acceso de los usuarios conforme a la UNEME en cuestión y programar las pruebas de acuerdo al número de equipos o material dependiendo del protocolo que se requiera evaluar. La responsabilidad de definir que protocolo será evaluado está a cargo del personal experto, además de que podrán monitorear el cumplimiento del registro de las pruebas.

El personal del **CNEGySR** mencionó que se requiere dar prioridad a las pruebas de los equipos digitales porque la UNEME-DEDICAM de Querétaro es el lugar donde se realizaran las pruebas piloto del programa para que una vez validada y aprobada la eficacia del software se procediera a su implementación a nivel federal.

Durante el desarrollo y la implementación de las pruebas, existieron varias adecuaciones a las mismas, lo anterior se aplicó tanto al manual como al desarrollo del software, esto porque algunas no cumplían con los objetivos del protocolo.

Lo anterior generó en consecuencia un aumento de tiempo en el desarrollo que implico la redefinición del software en más de una ocasión.

Capítulo IV – Desarrollo del "SCCEM"

1. ANÁLISIS

El análisis consta de tareas para descubrir la información, procesos, modelado y especificación de los requisitos que proporcionan el número de subprocesos a desarrollar, para diseñar y aportar una solución de software que podrá visualizar el usuario para valorarla.

1.1 INVESTIGACIÓN PRELIMINAR

En la investigación preliminar las actividades que se realizaron son:

1. Entrevistas con los participantes del proceso para entender y representar la información de manera simple para poder visualizar el dominio del problema.
2. Descripción y representación del comportamiento general del sistema de acuerdo a factores externos e internos por medio de la definición de funciones del software.
3. Agrupación de la información por medio de módulos de acuerdo a su comportamiento.
4. Definición de requerimientos generales del usuario y del software.
5. Planeación de actividades.

Actividad 1. Entrevistas con los participantes del proceso para entender y representar la información de manera simple para poder visualizar el dominio del problema.

Las entrevistas con el usuario se realizan para comprender y dominar la problemática, los usuarios describen y otorgan formatos acerca del proceso, además de que se logran acuerdos y se hacen propuestas por nuestra parte. En posteriores juntas se expone la comprensión de la problemática y su solución mediante el software, y también se hacen nuevos compromisos.

Los usuarios nos otorgaron formatos tanto en papel como en electrónico, los primeros solo hacen referencia a las hojas de registros de las pruebas, mientras el segundo formato hace referencia tanto a las hojas de registro de las pruebas como a las explicaciones de los protocolos. Durante y después de las entrevistas hicimos diferentes preguntas para aclarar algunas dudas, y poder comprender mejor la problemática, algunas preguntas fueron:

- ¿Qué tipo de dato o datos se van han de registrar?
- ¿Cuál es el número de hojas de registro total?
- ¿Para qué tipo de mastógrafos se clasificaron las pruebas?
- ¿Porqué algunas pruebas tiene valores de referencia y otras no?
- ¿Cuál es la frecuencia de aplicación de las hojas de registro?
- ¿Qué se califica en cada protocolo? y ¿por qué?
- ¿En cuántas unidades se van a aplicar estas hojas de registros?

- ¿Cuál es el número máximo de equipos en una unidad actualmente?
- ¿Cuál es el criterio para organizar la información de las hojas de registro?, y ¿Sí esta se podría reorganizar en la solución de software?
- ¿Se cuenta con una línea de acción ó proceso para la recolección de la información?
- ¿Todas las unidades cuentan con el mismo tipo de equipo?
- ¿Cuáles son las funciones de las personas que trabajan en la unidades?, ¿cuáles son las funciones que realizan? y ¿quiénes serían los encargados de registrar la información?, etc.

Lo primero es identificar el objetivo del proyecto de software y de acuerdo a ello posteriormente identificar cuál o cuáles serían los componentes relevantes que definan el comportamiento de la solución, estos son los puntos críticos o información significativa para definir el flujo de la información.

De acuerdo con la información otorgada identificamos que el objetivo del proyecto en lo general es replicar el **Manual de Control de Calidad en Mastografía**, y en lo particular está conformado por varios objetivos como realizar el registro de las pruebas de acuerdo al número de instrumentos a evaluar en cada una de las unidades, además se debe incluir un proceso de registro de las hojas a nivel federal.

Es importante resaltar que no solo se califican los mastógrafos sí no que existen otros instrumentos como son impresoras, cuartos de velo, chasises, monitores, y que también es posible crear la combinación de instrumentos para calificarlos en las hojas de registros por lo que el número de registros por prueba aumentará, lo que representa un factor crucial para el proceso de programación de las hojas de registro.

Para llegar a acuerdos con los usuarios redactamos una minuta para hacer constar que comprendimos la problemática en lo general, y considerando confidencial toda la información que el usuario nos otorgó.

MINUTA

Objetivo

Conocer los objetivos y alcances del proyecto así como cual es el proceso de trabajo y/o actividades que actualmente se realizan para la elaboración del mismo.

Temas tratados

1. **(AM)** y **(JR)** explicaron el proceso de control de calidad de mastografía que consiste en la aplicación de pruebas de control de calidad a los mastógrafos y equipos utilizados para la generación e interpretación de estudios de mastografía para verificar su correcto funcionamiento y asegurar la generación de imágenes mastográficas de alta calidad. El software que desarrollará la empresa SolSoft permitirá que las pruebas de control de calidad que debe realizar el personal técnico radiólogo puedan registrarse. Con esto se

podrá establecer un monitoreo constante e inmediato de los resultados de las pruebas que aportarán información sobre el estado de funcionamiento de los equipos de mastografía. Dicho monitoreo se realizará a los 253 centros que se encuentran repartidos en toda la republica mexicana.

2. **(AM), (JR), (AF) y (EM)** Se revisaron los siguientes formatos, cabe mencionar que el llenado de los mismos está especificado en el "Manual de control de calidad":

1. Procesador de Película

i. Esta prueba solo es para mastógrafos de tipo Analógico.
ii. La frecuencia de aplicación de la prueba es diaria.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.
iv. Para este formato se necesita que se definan los niveles de operación. Estos no cambian a menos que haya un ajuste en el equipo, cambio del mismo o por definición de los físicos médicos o por cambio de material. Para poder definir el nivel de operación se necesitan 5 días.
v. Las cotas máxima y mínima que indican los criterios de aceptación en las las gráficas se definen de acuerdo al nivel de operación.
vi. Para el nivel de aceptación se definió que los colores iban a ser Rojo para No Aceptable, Amarillo si se cayó en cota y verde para aceptable.
vii. Para obtener (MD) es el escalón cuyo promedio es cercano a 1.2 pero no menor a éste.
viii. Para obtener (LD) es el escalón cuyo promedio es cercano a 0.45 pero no menor a éste.
ix. Para obtener (HD) es el escalón cuyo promedio es cercano a 2.2, ya sea menor o mayor a éste.
x. Para obtener (DD) se tiene que hacer la resta de (HD) – (LD)
xi. Para obtener (B+V) siempre se toma el promedio del escalón 1.
xii. Para los valores vigentes, tienes que repetir los pasos anteriores, solo que es por un solo día.
xiii. Si al calcular la los valores vigentes, el MD, DD o B+V caen fuera de rango, hay que indicarle a la técnica que vuelva a realizar la prueba.
xiv. Es necesario graficar los valores de MD, DD y B+V con referencia de los niveles de operación. En el anexo A se puede ver un ejemplo de éstas gráficas.

2. Calidad de Imagen

i. Esta prueba solo es para mastógrafos de tipo Analógico.
ii. La frecuencia para la prueba es semanal.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es

necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.

iv. El registro de los parámetros: modo de exposición, Ánodo/Filtro, Posición del detector del CAE y Posición del control de densidad del CAE se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR.

v. Para tamaño de campo solo hay 2 opciones: 18x24 cm^2 o 24x30 cm^2

vi. Para Fibras, Motas y Masas se capturan números y el nivel de aceptación para cada uno es >= 4, >=3 y >=3.

vii. La técnica captura DO_F, DO_A y DO_D, y el sistema debe obtener DO_{DD} con la diferencia de DO_A – DO_D. Y el nivel de aceptación es si DO_F > 1.2 y si 0.35 >= DO_{DD} <= 0.45

viii. Para el nivel de aceptación se definió que aceptable va a ser verde y No Aceptable en rojo.

ix. Las cotas de las gráficas serían los rangos de aceptación.

x. Es necesario graficar los valores de DO_{DD}, DO_F, Fibras, Motas, Masas y %mAs el cuál se calcula con respecto al manual. En el anexo B se puede ver un ejemplo de éstas gráficas.

xi. En la captura del tamaño del campo, en caso de haber una observación se registra.

3. Cuarto Oscuro

i. Esta prueba solo es para mastógrafos de tipo Analógico.

ii. La frecuencia para la prueba es semestral.

iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.

iv. DO_C, DO_{NC} lo captura la técnica, y DO_V = DO_{NC} - DO_C.

v. Para el nivel de aceptación es DO_V <= 0.05 y se indica con un color verde que pinta la celda del resultado cuando éste es aceptable y se indica con un color rojo que pinta la celda cuando el resultado es no aceptable.

4. Contacto película-pantalla

i. Esta prueba solo es para mastógrafos de tipo Analógico.

ii. La frecuencia para la prueba es semestral.

iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.

iv. Se registra el Ánodo/Filtro (con catálogo), kVP y mAs.
v. Cuando el personal técnico radiólogo indique que existen regiones de contacto pobre se habilitará un campo para indicar qué región(es) es (son). De lo contrario, el resultado de la prueba es aceptable y debe indicarse con un color verde.

5. Evaluación de Fuerza de compresión

i. Esta prueba es común a todos los mastógrafos.
ii. La frecuencia para la prueba es mensual.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.
iv. El valor medido de fuerza de compresión que se indica en la báscula y la fuerza de compresión nominal se capturan por el personal técnico radiólogo y el resultado es sin signo.
v. Para el criterio de aceptación se toma la lectura de la báscula. Si es Aceptable se pone en verde, en caso contrario rojo.

6. Análisis de Estudios Repetidos

i. Esta prueba es común a todos los mastógrafos.
ii. La frecuencia para la prueba es trimestral.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.
iv. Para la opción de Fallas en revelado debe agregarse la leyenda No aplica para mastógrafos digitales (CR y DR).
v. Y para las opciones de Radiografía muy clara y Radiografía muy oscura cuando se trata de mastógrafos digitales se cambia la leyenda por Imagen muy clara e Imagen muy oscura respectivamente.
vi. Total de Repeticiones se calcula como Total de rechazos – las de Control de Calidad.
vii. Y los criterios de aceptación es que el % de Repetidas debe ser < al 2% del total de películas utilizadas en el periodo (verde), si es > al 5% del total de películas utilizadas del periodo (rojo) y si es >= al 2% hasta <= al 5% (amarillo).

7. Revisión visual del Equipamiento e Instalaciones

i. Esta prueba es común a todos los mastógrafos.
ii. La frecuencia para la prueba es mensual.

iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.
iv. Los datos que se capturan para el cuerpo del Mastógrafo y Panel de Mandos de acuerdo al siguiente código: Pasó (P), Falló (F) y No Aplica (NA).
v. Se indica con el código de colores verde o rojo cuando el resultado de la prueba es aceptable o no aceptable.

8. Análisis de Retención del Fijador en la Película

i. Esta prueba solo es para mastógrafos de tipo Analógico.
ii. La frecuencia para la prueba es trimestral.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie. Para el Fijador y Líquido para la prueba se capturan los datos anteriores.
iv. Se captura la cantidad de fijador residual 1 ó 2 donde 1 es aceptable y 2 no aceptable.
v. Se indica con el código de colores verde o rojo cuando el resultado de la prueba es aceptable o no aceptable.

9. Constancia de Funcionamiento Global del CAE

i. Esta prueba solo es para mastógrafos de tipo CR y DR.
ii. La frecuencia para la prueba es semanal.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.
iv. Para este formato se necesita que se definan los valores iniciales. Estos no cambian a menos que haya un ajuste en el equipo, cambio del mismo o por definición de los físicos médicos.
v. El personal técnico radiólogo captura (de catálogo) ánodo/filtro, kVp, mAs, VMP, DTP. El software calculará RSR y RCR automáticamente guardará los valores iniciales. Posteriormente el registro y comparación de estos mismos datos se hará con la periodicidad indicada.
vi. Una vez que se tienen los valores iniciales ya es posible calcular los valores vigentes, donde RSR se calcula de la misma forma y RCR se calcula con la fórmula del “Manual de calidad”.
vii. El porcentaje de discrepancia se calcula como $((Valor_i - Valor_{vigente})/Valor_i)*100$

viii. Se grafica la discrepancia de mAs, discrepancia en RSR y discrepancia en RCR, y se hace para 3 espesores de acrílico que son 2.0 cm, 5.0 cm y 6.0 cm.
ix. Se indica con el código de colores verde o rojo cuando el resultado de la prueba es aceptable o no aceptable.

10. Constancia en la Homogeneidad del Receptor

i. Esta prueba solo es para mastógrafos de tipo CR y DR.
ii. La frecuencia para la prueba es semanal.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.
iv. Al formato le hace falta definir el patrón de homogeneidad a evaluar (solo hay 3, consultar el "Manual de control de calidad"), y dependiendo de esto se habilitan la cantidad de regiones de interés. Para trabajar valores iniciales así como valores vigentes.
v. Para este formato se necesita que se definan los valores iniciales. Estos no cambian a menos que haya un ajuste en el equipo, cambio del mismo o por definición de los físicos médicos y dependen del patrón. Los valores iniciales o niveles de operación se establecen cuando se practica por primera vez la prueba y el software los almacena como tal.
vi. Una vez que se tienen los valores iniciales ya es posible capturar los niveles de operación.
vii. Para los valores vigentes dependiendo del patrón la técnica sólo captura VMP y DTP, el sistema debe calcular automáticamente RSR, RSR promedio etc.
viii. Para obtener el %Mínimo, se usa $((Valor_{mínimo} – Valor_{promedio})/Valor_{promedio})*100$ y para obtener el %Máximo, se usa $((Valor_{máximo} – Valor_{promedio})/Valor_{promedio})*100$
ix. El sistema debe llenar en donde dice criterio de aceptación con el cálculo del 5% del %Mínimo inicial y %Máximo inicial.
x. El criterio de aceptación se da si él % Máximo del valor vigente y él % Mínimo del valor vigente no sobrepasan el criterio de aceptación calculado previamente. Se pone verde para aceptable y rojo para no aceptable.

11. Artefactos en mastógrafos DR y CR

i. Esta prueba solo es para mastógrafos de tipo CR y DR.
ii. La frecuencia para la prueba es semanal.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie. La rejilla de contacto rejilla sólo aplica para mastógrafos DR recuerden que en esta prueba existen 2 formatos, uno para CR y otro para DR.

iv. Para este formato se necesita que se definan los valores iniciales. Estos no cambian a menos que haya un ajuste en el equipo, cambio del mismo o por definición de los físicos médicos.
v. Los valores iniciales se establecen cuando se aplica por primera vez la prueba o cuando se indica al software que se restablecerán los valores iniciales (por reparaciones en los equipos u otros motivos) y el software debe registrarlos y mantenerlos como referencia; una vez que se tienen los valores iniciales ya es posible capturar los niveles de operación.
vi. Para los valores vigentes la técnica captura el Ancho de ventana y el Alto de ventana y especifica si uso alguna herramienta de visualización. Si selecciona Otra como herramienta, debe capturar cual.
vii. La técnica captura o indica los artefactos visualizados. Si tiene comentarios los puede capturar.
viii. Al final de los artefactos se agrega un renglón de aceptación y se considera como aceptable cuando todo tiene palomita y se pone en verde, de caso contrario se pone en rojo.
ix. En el caso de los mastógrafos CR se hace un formato aparte y lo que cambia son las condiciones clínicas.

12. Constancia de la Calidad de la Imagen

i. Esta prueba solo es para mastógrafos de tipo CR y DR.
ii. La frecuencia para la prueba es quincenal.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.
iv. Para este formato se necesita que se definan los valores iniciales. Estos no cambian a menos que haya un ajuste en el equipo, cambio del mismo o por definición de los físicos médicos. Los valores iniciales se deben tomar para la Estación de trabajo, Estación de adquisición e Imagen impresa.
v. Una vez que se tienen los valores iniciales ya es posible capturar las condiciones clínicas. Para que el Ánodo/Filtro sea aceptable, tiene que ser igual que el de los valores iniciales. Verde para aceptable y rojo para no aceptable.
vi. Para los valores vigentes la técnica captura las fibras, el grupo de motas y las masas para Estación de trabajo, Estación de adquisición e Imagen impresa.
vii. Si los valores vigentes son mayores o iguales a los valores iniciales, entonces se consideran como aceptables, usando el código de colores definido.
viii. Se requiere graficar los valores de Fibras, Motas y Masas, así como obtener la Discrepancia en mAs (como se indica en el Manual de calidad y ésta se grafica).

13. Evaluación de monitores

i. Esta prueba solo es para mastógrafos de tipo CR y DR.

ii. La frecuencia para la prueba es semanal.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie. Estos dos son opcionales. Este último monitor si es obligatorio ya que todos los centros cuentan con al menos uno.
iv. La técnica captura la Iluminancia ambiental medida y si es <= 10 entonces el valor es aceptable (verde) de otra forma es no aceptable (rojo).
v. La técnica tiene que marcar si los elementos a evaluar son o no aceptable
vi. En la parte final de esta lista se agrega la fila que dice ¿Aceptable?, si 4 o mas elementos son aceptables, entonces la prueba se considera como aceptable (verde), de otra forma es no aceptable (rojo).
vii. También para medir la Luminancia la técnica debe capturar la Luminancia máxima y mínima para el monitor Izquierdo y derecho (si tienen) y el de Adquisición (obligatorio). Si la $L_{máx}$ es >= 170 (monitor derecho), >= 170 (monitor izquierdo) y >= 100 (Adquisición), entonces se considera como aceptable. Después se obtiene otra columna de $L_{máx}/L_{mín}$, si esta columna es >= 250 (monitor derecho), >= 250 (monitor izquierdo) y >= 100 (adquisición), entonces se considera como aceptable. Si los dos niveles de aceptación son aceptables, entonces se considera la prueba como aceptable y se pinta de verde, en caso contrario es no aceptable y se pinta de rojo.
viii. Para la parte de homogeneidad la técnica captura los valores de L_c, L_{si}, L_{sd}, L_{ii} y L_{id}. El sistema toma los valores máximos y mínimos para definir $L_{máx}$ y $L_{mín}$ y con estos calcula $(L_{máx} – L_{mín})/L_c$, si el resultado de éste es < 0.3 para el caso de monitor izquierdo, derecho y de adquisición, entonces las pruebas se consideran como aceptables (verde), de otra forma son no aceptables (rojo).

14. Evaluación de Impresoras

i. Esta prueba solo es para mastógrafos de tipo CR y DR.
ii. La frecuencia para la prueba es semanal.
iii. El registro de las marcas y modelos de los equipos de mastografía y equipos de medición se realizará a través de la selección de una lista extraída de catálogos proporcionados por AM y JR. Para los números de serie es necesario que la técnica los capture, aunque la captura sólo debe ser inicial para que el personal técnico radiólogo no teclee por siempre los números de serie.
iv. La técnica tiene que marcar si los elementos a evaluar son o no aceptables
v. En la parte final de esta lista se agrega la fila que dice ¿Aceptable?, si 4 o más elementos son aceptables, entonces la prueba se considera como aceptable (verde), de otra forma es no aceptable (rojo).
vi. Para los niveles extremos de densidad óptica la técnica captura $DO_{mín}$ y $DO_{máx}$, y tienen que ser < 0.25 y > 3.4 respectivamente para ser aceptables, si los dos valores son aceptables entonces la prueba es aceptable (verde), de lo contrario es no aceptable (rojo).

vii. Para la parte de homogeneidad de la imagen impresa, la técnica captura los valores de DO_c, DO_{si}, DO_{sd}, DO_{ii} y DO_{id}. El sistema toma los valores máximos y mínimos para definir $DO_{máx}$ y $DO_{mín}$ y con estos calcula $(DO_{máx} – DO_{mín})/((DO_{máx} + DO_{mín})/2)$, si el resultado de éste es < 0.1 entonces es aceptable (verde), de lo contrario no aceptable (rojo).

viii. Para los valores de sensitometría es necesario tener una tabla similar a la de procesador de película pero con 30 escalones y un solo día y sin $DO_{promedio}$. Con esto obtenemos como valores iniciales Dmax, HD, LD, DD=HD-LD, Dmedia y B+V.

ix. Después definida los valores iniciales, entonces obtenemos la Dmax, HD, LD, DD=HD-LD, Dmedia y B+V vigentes y si cumplen los valores de aceptación, entonces se considera que la prueba fue Aceptable (verde) y en caso contrario No aceptable (rojo).

x. Los valores iniciales dan las cotas de las gráficas y se grafica D_{max}, DD, D_{Media} y B+V.

Las minutas son un respaldo documental de los temas o acuerdos tratados en las reuniones de trabajo y fueron utilizadas para exponer la comprensión de la problemática con nuestras ideas o a nuestra manera y en caso de tener dudas poder consultarlas posteriormente dado que el usuario reviso y valido el contenido de las mismas. De esta forma también podemos resaltar todos los puntos que se deben abordar dentro de la solución de software. Por ejemplo, hay ocasiones en que los usuarios omiten la información importante y sí posteriormente quieren cambiar los acuerdos, es en estos casos en los que una minuta se puede utilizar para darnos cuenta de cuáles fueron los acuerdos originales y en caso de aplicar cambios que sean de común acuerdo.

Para identificar el detalle de los requerimientos fue necesario revisar en varias ocasiones los formatos e ir cuestionando más a fondo a los usuarios ya que al principio solo nos platicaron el proceso en forma muy general, y esto no permite visualizar todos los componentes del problema.

Durante las entrevistas también fue importante considerar las expectativas de software por parte de los usuarios.

Actividad 2. Descripción y representación del comportamiento general del sistema de acuerdo a factores externos e internos por medio de la definición de funciones del software.

Existen diferentes formas de comprensión de la problemática:

- Diagramando procesos.
- Identificando puntos críticos tanto de procesos como del flujo de la información.
- Identificando prioridades y dependencias de las actividades dentro de un proceso.
- Enlistar subprocesos.
- Vislumbrar roles involucrados y su corresponsabilidad en las actividades.

Para descubrir el comportamiento de la solución de software es necesario detectar puntos críticos o de riesgos porque definen y dan orientación al comportamiento de la solución, sirviéndonos de apoyo el haber elaborado las minutas, realizar preguntas a los usuarios, así como clasificar la información que se nos proporcionó las veces que fueran necesario.

Algunos de los puntos críticos que se detectaron son:

- Lo que se califica en cada una de las hojas de registro de las pruebas.
- El número de unidades donde se tendrá que replicar.
- La frecuencia de programación de las hojas de registro.
- Programación de las hojas de registro de las pruebas.

Una manera en la que se muestra de forma simple cuales son los roles involucrados, la infraestructura, así como el flujo de la información que formará parte de la solución es por medio de un esquema general.

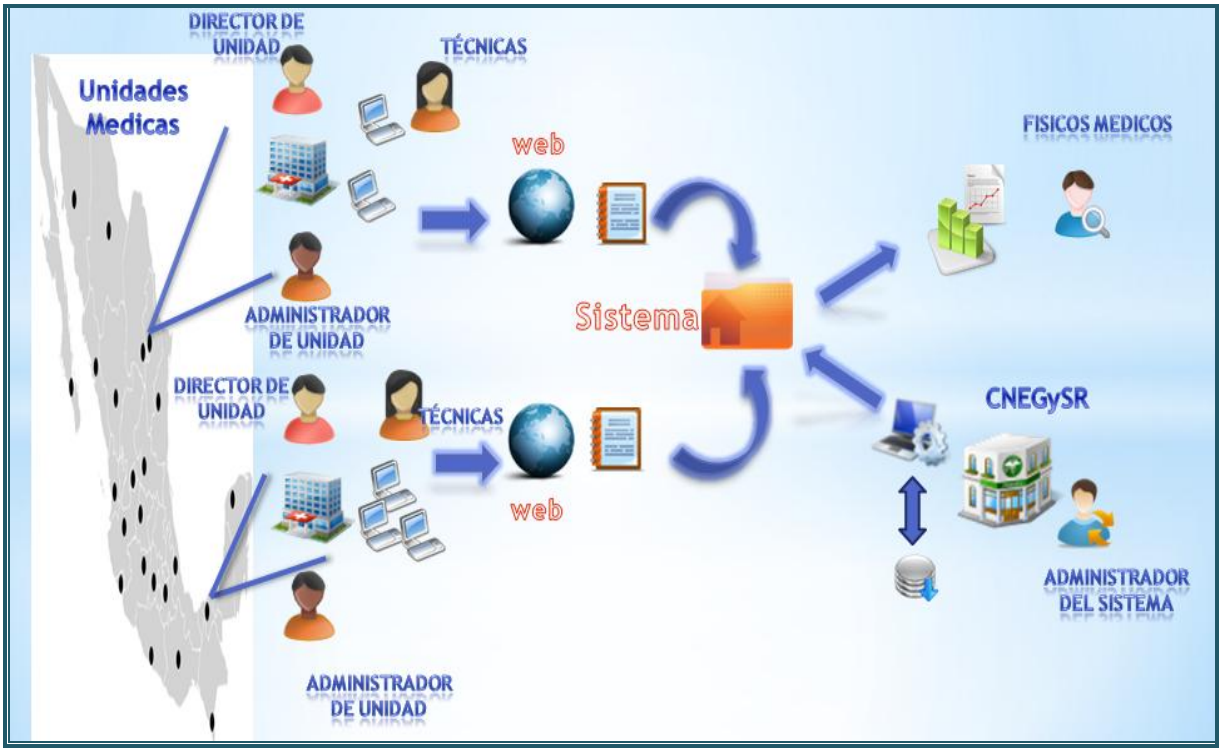


Figura 4.1 ***Esquema general de la solución propuesta.***[2]

Este diagrama nos da una panorámica de la solución de software y de los factores internos y externos que posiblemente pueden existir como:

- Factores internos
  - Definición de roles.
  - Módulos del software.

[2] Planteamiento general de los steel holders.

- Factores externos
    - Infraestructura en las unidades.
    - Infraestructura del **CNEGySR**.
    - Líneas de acción para el proceso de captura.

Actividad 3. Agrupación de la información por medio de módulos de acuerdo a su función.

Posteriormente clasificamos la información de los formatos, de acuerdo al tipo de dato, al tipo de instrumento, material o equipo, a la frecuencia de programación, también identificamos sí se necesitan valores de referencia, graficas, y que datos son los que se grafican, ver si los criterios de aceptación de cada una de las pruebas se pueden homogeneizar.

Con la información otorgada y la comprensión de la problemática empezamos a elaborar un listado de requerimientos y estos los clasificamos por módulos ya que cada uno de ellos nos indica el comportamiento de acuerdo al flujo de la información. También clasificamos la información de acuerdo con el enfoque que decidimos darle a la solución.

Mapa mental para orientar la solución de software (visión general).

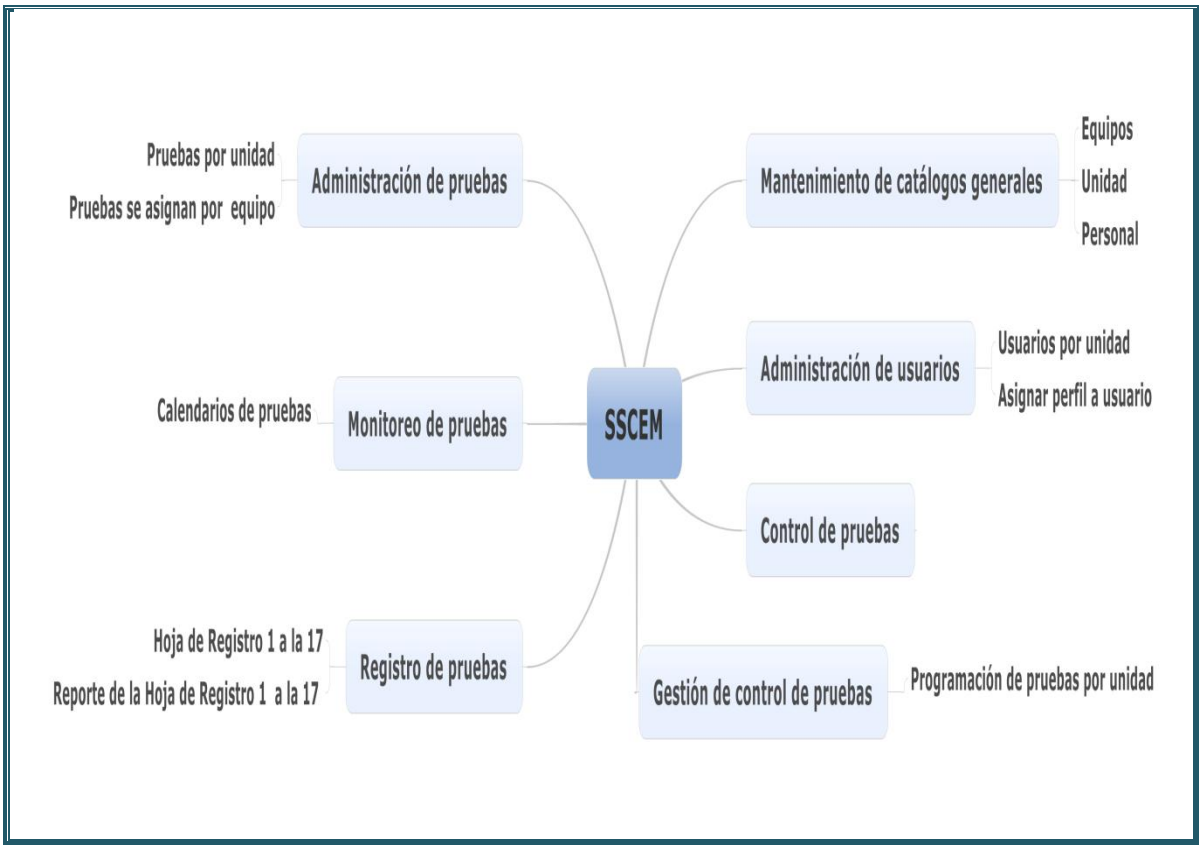


Figura 4.2 ***Mapa mental general de la solución planteada.***

Mapa mental para orientar la solución de software (visión detallada).

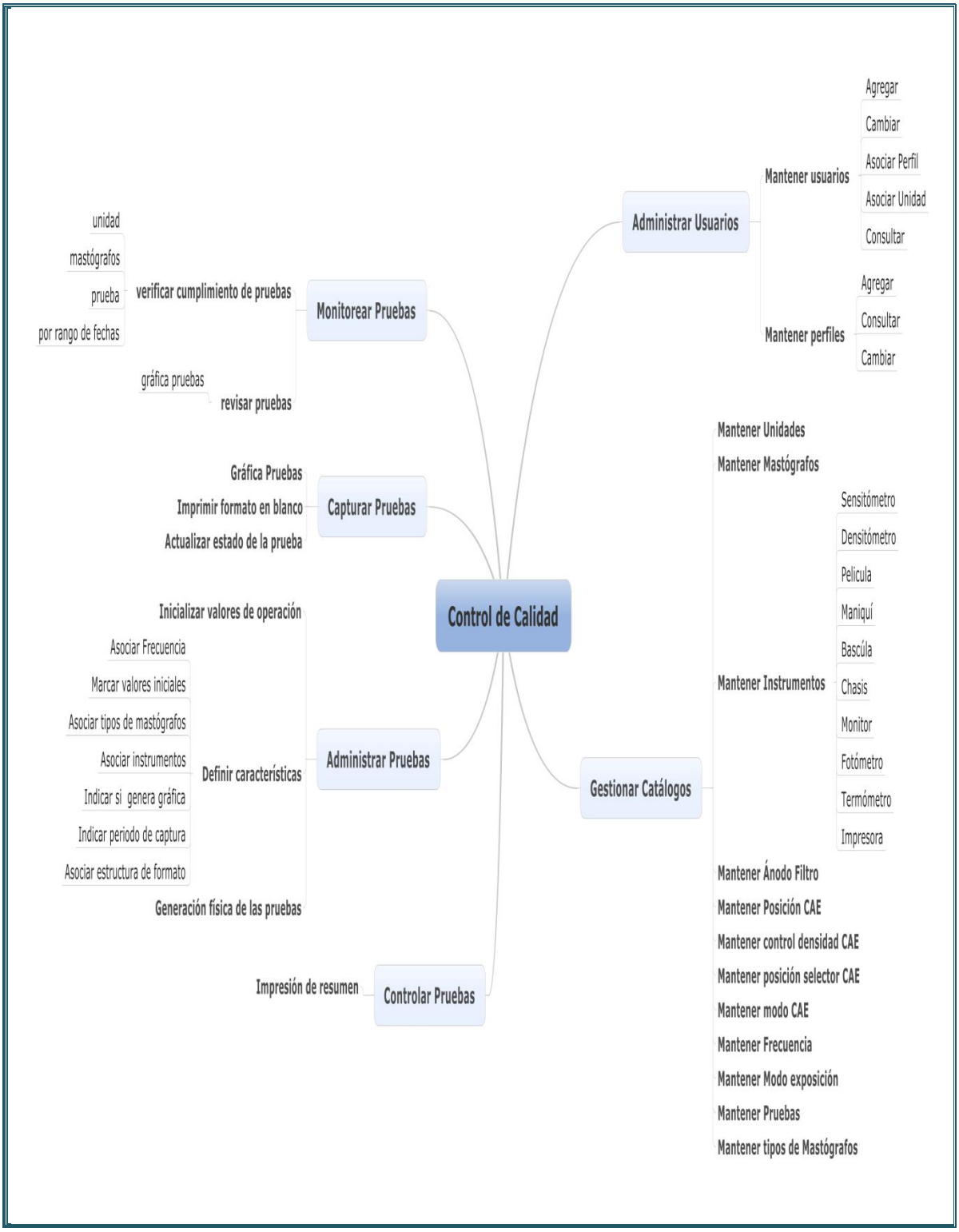


Figura 4.2 ***Mapa mental detallado de la solución planteada.***

Actividad 4. Definición de requerimientos generales del usuario y del software

Con la clasificación de la información pudimos darnos cuenta del número de requerimientos que nos solicitaron, y en las entrevistas consideramos otras necesidades más generales o particulares para la organización de la solución de software.

Por ejemplo a continuación se muestra una de las primeras clasificaciones de la información.

PRUEBAS					GENERALIDADES					EQUIPO					INSTRUMENTOS								MATERIAL				
#	Descripción	T. Mastógrafo	Frecuencia	Etapa	Formato pdf	Formato de captura	Pantalla de búsqueda	Nivel operación	Gráfica	Mastógrafo	Procesador	Monitor	Impresora Especializada	Estación de Adquisición	Densitómetro	Sensitómetro	Maniquí	Chasis	Bascula	Rejilla de Contacto	Fotómetro	Termómetro	Película	Fijador	Líquido para la prueba	Revelador	Imagen
F1.A	Hoja de Registro del procesador de película 1	analógico	diaria	2																							
F1.B	Hoja de Registro del procesador de película 2	analógico	diaria	2																							
F1.C	Carta de Control 1 del procesador	analógico	diaria	2																							
F1.D	Hoja de Registro del procesador de película	analógico	diaria	2																							
F1.F	Transición entre cajas de películas	analógico	diaria	2																							
F2	Calidad de la Imagen	analógico	semanal	2																							
F3	Hoja de Registro del cuarto Oscuro	analógico	semestral	2																							
F4	Contacto Película-Pantalla	analógico	semestral	2																							
F5	Evaluación de Fuerza de Compresión	común	mensual	*1*																							
F6	Análisis de Estudio Repetido	común	trimestral	*1*																							
F7	Revisión Visual de Equipamiento e Instalaciones	común	mensual	*1*																							
F8	Hoja de Registro de Análisis de Retención del Fijador de película	analógico	trimestral	2																							
F9	Constancia del Funcionamiento Global del CAE	digital	semanal	*1*																							
F10	Constancia en la Homogeneidad del Receptor	digital	semanal	*1*																							
F11.A	Artefactos en Mastógrafos DR	digital	semanal	*1*																							
F11.B	Artefactos en Mastógrafos CR	digital	semanal	*1*																							
F12	Constancia de la Calidad de la Imagen	digital	quincenal	*1*																							
F13	Evaluación de Monitores	digital	semanal	*1*																							
F14	Evaluación de Impresoras	digital	semanal	*1*																							

Requerimientos[3]

Con lo anterior clasificación podemos empezar a listar algunos requerimientos generales como:

- El total de hojas de registros de acuerdo al tipo de mastógrafos para el cual el protocolo está diseñado.
- ¿Cuáles cuentan con valores de referencia?
- ¿Qué instrumento califica cada protocolo?
- Control de acceso al sistema

[3] Matriz de Requerimientos para la identificación de parámetros que son determinantes para el planteamiento de la solución propuesta.

- Administración de usuarios.
- Que el software esté disponible en web.
- Monitoreo de pruebas por parte del personal experto.
- Control de pruebas.
- Registro de las Pruebas.
- Gráficas
- Reporte de las hojas de registros
- Niveles de operación para algunas hojas de registro
- Administración de catálogos
- Programación de las pruebas de acuerdo a su frecuencia.
- Definición de las pruebas por unidad.
- Que las hojas solo se puedan registrar en dos ocasiones una vez programada sin importar que criterio de aceptación se obtuvo.
- Lenguaje de programación
- Un servidor web
- Un manejador de base de datos
- Considerar el crecimiento de la base de datos

Actividad 5. Planeación de actividades

La planeación de actividades consiste en organizarnos con el objetivo de dar mejores resultados en el proyecto, identificando cuales cuantas son las actividades que formaran el plan de trabajo, algunas de estas actividades son:

- Generar el listado de requerimientos particulares.
- Asignación de actividades.
- Descripción de casos de uso.
- Diseño del sistema.
- Seleccionar el lenguaje de programación.
- Seleccionar el manejador de base de datos.
- Seleccionar el servidor web.
- Considerar el crecimiento de la base de datos
- Calendarización o cronograma
- Seguimientos de las actividades

De acuerdo a los roles de nuestro equipo de trabajo se designó a un responsable de la asignación las actividades, del seguimiento, del cumplimiento así como de resolver eventualidades, generar acuerdos.

1.2 ANÁLISIS DE LAS NECESIDADES DEL SISTEMA

En el análisis de las necesidades del sistema las actividades que se realizaron son:

1. Lista detallada de requerimientos.
2. Cronograma.
3. Casos de uso de los módulos propuestos.

Actividad 1. Lista detallada de requerimientos.

Los requerimientos se van vislumbrando de acuerdo a las necesidades de usuarios. Cada uno de ellos responde a un paso dentro de una serie de actividades para cumplir con una tarea. Es importante visualizar cuantas tareas o actividades se van a realizar para considerar el tiempo necesario para desarrollar el proyecto, además de que con esto podemos calcular la cantidad de recursos que se requerirán.

Con el análisis de la información proporcionado por medio de las entrevistas con los usuarios, minutas, clasificación de la información, de varias reuniones de trabajo llegamos a elaborar un listado más detallado de los requerimientos:

Categoría	Descripción	Tipo
Control de Acceso	Pantalla de control de acceso	Pantalla
Control de acceso	Proceso de verificación de usuario	Proceso
Administración de usuarios	Pantalla de mantenimiento de usuarios	Pantalla
Administración de usuarios	Asociar perfil al usuario	Pantalla
Administración de usuarios	Asociar unidad al usuario	Pantalla
Administración de usuarios	Pantalla de mantenimiento de perfiles	Pantalla
Catálogos	Pantalla de mantenimiento de unidades	Pantalla
Catálogos	Asociación de mastógrafos a las unidades	Proceso
Catálogos	Asociar Chasis a Mastógrafos	Proceso
Catálogos	Asociar Tipo de Mastógrafos a Mastógrafos	Proceso
Catálogos	Pantalla de mantenimiento Marcas	Pantalla
Catálogos	Pantalla de mantenimiento Modelos	Pantalla
Catálogos	Pantalla de mantenimiento de Instrumentos	Pantalla
Catálogos	Pantalla de mantenimiento de Ánodo Filtro	Pantalla
Catálogos	Pantalla de mantenimiento Posición CAE	Pantalla
Catálogos	Pantalla de mantenimiento Posición control de densidad CAE	Pantalla
Catálogos	Pantalla de mantenimiento Modo CAE	Pantalla
Catálogos	Pantalla de mantenimiento Posición del detector del CAE	Pantalla
Catálogos	Pantalla de mantenimiento Posición del selector de densidad del CAE	Pantalla
Catálogos	Pantalla de mantenimiento Modo de exposición	Pantalla

Catálogos	Pantalla de mantenimiento Frecuencia	Pantalla
Catálogos	Pantalla de mantenimiento Pruebas	Pantalla
Catálogos	Pantalla de mantenimiento Tipos de Mastógrafos	Pantalla
Control de pruebas	Pantalla de control de pruebas	Pantalla
Control de pruebas	Reporte de Resumen de control	Reporte
Control de pruebas	Actualizar estado de las pruebas	Proceso
Administración de pruebas	Pantalla de menú de administración de pruebas	Pantalla
Administración de pruebas	Pantalla para administrar pruebas	Pantalla
Administración de pruebas	Pantalla de inicialización de valores operacionales	Pantalla
Administración de pruebas	Pantalla para definir las características de las pruebas	Pantalla
Administración de pruebas	Asociar frecuencia a pruebas	Pantalla
Administración de pruebas	Asociar valores iniciales a pruebas	Pantalla
Administración de pruebas	Asociar tipo de mastógrafos a pruebas	Pantalla
Administración de pruebas	Asociar mastógrafos a pruebas	Pantalla
Administración de pruebas	Asociar estructura de formatos a pruebas	Pantalla
Administración de pruebas	Asociar instrumentos	Pantalla
Administración de pruebas	Indicar si se grafica a prueba	Pantalla
Administración de pruebas	Indicar periodo de captura a prueba	Pantalla
Generación física de las pruebas	Pantalla de generación de pruebas	Pantalla
Generación física de las pruebas	Proceso de generación de pruebas	Proceso
Monitorear pruebas	Menú del monitoreo de pruebas	Pantalla
Monitorear pruebas	Pantalla para revisar pruebas	Pantalla
Monitorear pruebas	Pantalla para mostrar graficas de las pruebas	Pantalla
Monitorear pruebas	Pantalla de cumplimiento de pruebas	Pantalla
Monitorear pruebas	Proceso de búsqueda por unidad	Proceso
Monitorear pruebas	Proceso de búsqueda por Mastógrafos	Proceso
Monitorear pruebas	Proceso de búsqueda por prueba	Proceso
Monitorear pruebas	Proceso de búsqueda por rango de fechas	Proceso
Capturar pruebas	Pantalla de Procesador de Película	Pantalla
Capturar pruebas	Pantalla de Calidad de Imagen	Pantalla
Capturar pruebas	Pantalla de Cuarto Oscuro	Pantalla
Capturar pruebas	Pantalla de Contacto película-pantalla	Pantalla
Capturar pruebas	Pantalla de Evaluación de Fuerza de compresión	Pantalla
Capturar pruebas	Pantalla de Análisis de Estudios Repetidos	Pantalla

Capturar pruebas	Pantalla de Revisión visual del Equipamiento e Instalaciones	Pantalla
Capturar pruebas	Pantalla de Análisis de Retención del Fijador en la Película	Pantalla
Capturar pruebas	Pantalla de Constancia de Funcionamiento Global del CAE	Pantalla
Capturar pruebas	Pantalla de Constancia en la Homogeneidad del Receptor	Pantalla
Capturar pruebas	Pantalla de Artefactos en mastógrafos DR y CR	Pantalla
Capturar pruebas	Pantalla de Constancia de la Calidad de la Imagen	Pantalla
Capturar pruebas	Pantalla de Evaluación de Monitores	Pantalla
Capturar pruebas	Pantalla de Evaluación de Impresoras	Pantalla
Capturar pruebas	Impresión de Procesador de Película	Reporte
Capturar pruebas	Impresión de Calidad de Imagen	Reporte
Capturar pruebas	Impresión de Cuarto Oscuro	Reporte
Capturar pruebas	Impresión de Contacto película-Impresión	Reporte
Capturar pruebas	Impresión de Evaluación de Fuerza de compresión	Reporte
Capturar pruebas	Impresión de Análisis de Estudios Repetidos	Reporte
Capturar pruebas	Impresión de Revisión visual del Equipamiento e Instalaciones	Reporte
Capturar pruebas	Impresión de Análisis de Retención del Fijador en la Película	Reporte
Capturar pruebas	Impresión de Constancia de Funcionamiento Global del CAE	Reporte
Capturar pruebas	Impresión de Constancia en la Homogeneidad del Receptor	Reporte
Capturar pruebas	Impresión de Artefactos en mastógrafos DR y CR	Reporte
Capturar pruebas	Impresión de Constancia de la Calidad de la Imagen	Reporte
Capturar pruebas	Impresión de Evaluación de Monitores	Reporte
Capturar pruebas	Impresión de Evaluación de Impresoras	Reporte
Capturar pruebas	Actualizar estado de la prueba	Proceso
Capturar pruebas	Graficar pruebas de Procesador de Película	Pantalla
Capturar pruebas	Graficar pruebas de Calidad de Imagen	Pantalla
Capturar pruebas	Graficar pruebas de Constancia de Funcionamiento Global del CAE	Pantalla
Capturar pruebas	Graficar pruebas de Constancia de la Calidad de la Imagen	Pantalla
Capturar pruebas	Graficar pruebas de Evaluación de Impresoras	Pantalla

Listado de Funciones[4]

Esta es lista final de requerimientos misma que fue objeto de varias modificaciones durante el desarrollo de la solución del software.

Actividad 2.Cronograma.

Estos requerimientos se calendarizan y se les asigna un responsable y posteriormente se supervisa el avance de los mismos dependiendo de los acuerdos que se realizaron con los usuarios.

Con la calendarización se pretende evitar

1. Perder el objetivo de la solución
2. Desmotivación del equipo
3. Vislumbrar otras oportunidades

[4] Se muestra el listado completo de las funciones realizadas de para la solución planteada del SCCEM.

Pueden existir diversos factores que retrasen el proyecto por ésta razón es importante que el líder de proyecto pueda resolver eventualidades contando con el apoyo de su equipo de trabajo, tomando acciones y decisiones concretas.

Por otro lado con el cronograma podemos ver que tareas son dependientes de otras, cuales implican mayor o menor tiempo, los tiempos para la implementación, tiempos de pruebas obligatorios ya sea por módulos o por cada una de las tareas.

En nuestro caso el calendario se replanteó por diferentes eventualidades tales como la redefinición de los protocolos, o por cambio de prioridades por parte de los usuarios.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predece	Nombres de los recursos
1		**Monitoreo de Indicadores**	**96 días**	**mié 13/06/12**	**mié 24/10/12**		
2		**Análisis y Diseño**	**4 días**	**mié 13/06/12**	**lun 18/06/12**		
3		Entrevistas con usuarios	1 día	mié 13/06/12	mié 13/06/12		
4		Diagrama de casos de uso	1 día	vie 15/06/12	vie 15/06/12	3	
5		Diagrama entidad/relación	1 día	lun 18/06/12	lun 18/06/12	4	
6		**Catálogos**	**7 días**	**lun 25/06/12**	**mar 03/07/12**		
7		Pantalla de mantenimiento de entidad federativa	1 día	lun 25/06/12	lun 25/06/12		AFP
8		Pantalla de mantenimiento de Unidad Médica	1 día	lun 25/06/12	lun 25/06/12	7	AFP
9		Pantalla de mantenimiento de Prueba	1 día	mar 26/06/12	mar 26/06/12	8	AFP
10		Pantalla de mantenimiento de medida de tiempo	1 día	mar 26/06/12	mar 26/06/12	9	AFP
11		Pantalla de mantenimiento de frecuencia	1 día	mar 26/06/12	mar 26/06/12	10	AFP
12		Pantalla de mantenimiento de tipo de instrumento	1 día	mié 27/06/12	mié 27/06/12	11	AFP
13		Pantalla de mantenimiento de Instrumento	1 día	jue 28/06/12	jue 28/06/12	12	AFP
14		Pantalla de mantenimiento Modo de exposición	1 día	vie 29/06/12	vie 29/06/12	13	AFP
15		Pantalla de mantenimiento de tipo de mastografo	1 día	lun 25/06/12	lun 25/06/12		EMM
16		Pantalla de mantenimiento de ánodo/filtro	1 día	lun 25/06/12	lun 25/06/12	15	EMM
17		Asociación de tipos mastógrafo con ánodo/filtro	1 día	mar 26/06/12	mar 26/06/12	16	EMM
18		Pantalla de mantenimiento de marca	1 día	mié 27/06/12	mié 27/06/12	17	EMM
19		Pantalla de mantenimiento de modelo	1 día	jue 28/06/12	jue 28/06/12	18	EMM
20		Asociación de tipos de instrumento con marcas	1 día	vie 29/06/12	vie 29/06/12	19	EMM
21		Pantalla de mantenimiento de tipo de programación	1 día	lun 02/07/12	lun 02/07/12	20	EMM
22		Pantalla de mantenimiento Posición del detector del CAE	1 día	lun 02/07/12	lun 02/07/12	21	EMM
23		Pantalla de mantenimiento Posición del selector de densidad del CAE	1 día	mar 03/07/12	mar 03/07/12	22	EMM
24		**Gestión de pruebas**	**1 día**	**lun 25/06/12**	**lun 25/06/12**		
25		Pantalla de definición de pruebas	2 días	lun 02/07/12	mar 03/07/12	14	AFP
26		Pantalla de asociaciación de definición de pruebas con unidades médicas	2 días	mié 04/07/12	jue 05/07/12	25	AFP
27		Pantalla de asociación de instrumentos con definición de prueba	2 días	vie 06/07/12	lun 09/07/12	26	AFP
28		Proceso de programación de la prueba	2 días	mar 10/07/12	mié 11/07/12	27	AFP
29		Reporte de Resumen de control	3 días	lun 25/06/12	mié 27/06/12		
30		Actualizar estado de las pruebas	1 día	jue 28/06/12	jue 28/06/12	29	
31		**Monitorear pruebas**	**1 día**	**jue 12/07/12**	**jue 12/07/12**		
32		Pantalla de generación de pruebas	2 días	jue 12/07/12	vie 13/07/12	28	AFP
33		Proceso de generación de pruebas	4 días	lun 16/07/12	jue 19/07/12	32	AFP
34		Pantalla de calendario de monitoreo de pruebas	6 días	vie 20/07/12	vie 27/07/12		
35		**Capturar pruebas**	**59 días**	**mié 04/07/12**	**lun 24/09/12**		
36		Cabecera de pruebas	4 días	mié 04/07/12	lun 09/07/12	23	EMM
37		Pantalla de Evaluación de Fuerza de compresión	5 días	lun 09/07/12	vie 13/07/12	36	EMM
38		Pantalla de Análisis de Estudios Repetidos	5 días	vie 13/07/12	jue 19/07/12	37	EMM
39		Pantalla de Revisión visual del Equipamiento e Instalaciones	5 días	jue 19/07/12	mié 25/07/12	38	EMM
40		Pantalla de Constancia en la Homogeneidad del Receptor	3 días	lun 30/07/12	mié 01/08/12	39	EMM
41		Pantalla de Constancia de Funcionamiento Global del CAE	3 días	jue 02/08/12	lun 06/08/12	40	EMM
42		Pantalla de Artefactos en mastógrafos DR y CR	5 días	jue 02/08/12	mié 08/08/12	41	EMM
43		Pantalla de Evaluación de Monitores	3 días	lun 30/07/12	mié 01/08/12	34	EMM
44		Pantalla de Evaluación de Impresoras	4 días	jue 02/08/12	mar 07/08/12	43	EMM
45		Pantalla de Constancia de la Calidad de la Imagen	4 días	mié 08/08/12	lun 13/08/12	44	EMM
46		Pantalla de Procesador de Película	6 días	mié 08/08/12	mié 15/08/12	42	AFP
47		Pantalla de Calidad de Imagen	4 días	mié 15/08/12	lun 20/08/12	46	EMM
48		Pantalla de Cuarto Oscuro	4 días	lun 20/08/12	jue 23/08/12	47	AFP
49		Pantalla de Contacto película-pantalla	3 días	mar 14/08/12	jue 16/08/12	45	AFP
50		Pantalla de Análisis de Retención del Fijador en la Película	6 días	mar 14/08/12	mar 21/08/12	49	EMM
51		Impresión de Evaluación de Fuerza de compresión	3 días	mié 22/08/12	vie 24/08/12	50	AFP
52		Impresión de Análisis de Estudios Repetidos	4 días	jue 23/08/12	mar 28/08/12	48	AFP
53		Impresión de Revisión visual del Equipamiento e Instalaciones	4 días	mar 28/08/12	vie 31/08/12	52	AFP
54		Impresión de Análisis de Retención del Fijador en la Película	4 días	vie 31/08/12	mié 05/09/12	53	EMM
55		Impresión de Constancia de Funcionamiento Global del CAE	4 días	mié 05/09/12	lun 10/09/12	54	EMM
56		Impresión de Constancia en la Homogeneidad del Receptor	3 días	lun 27/08/12	mié 29/08/12	51	AFP
57		Impresión de Artefactos en mastógrafos DR y CR	3 días	jue 30/08/12	lun 03/09/12	56	EMM
58		Impresión de Constancia de la Calidad de la Imagen	3 días	mar 04/09/12	jue 06/09/12	57	EMM
59		Impresión de Evaluación de Monitores	3 días	vie 07/09/12	mar 11/09/12	58	EMM
60		Impresión de Evaluación de Impresoras	4 días	lun 10/09/12	jue 13/09/12	55	AFP
61		Graficar pruebas de Procesador de Película	4 días	jue 13/09/12	mar 18/09/12	60	AFP
62		Graficar pruebas de Calidad de Imagen	4 días	mar 18/09/12	vie 21/09/12	61	EMM
63		Graficar pruebas de Constancia de Funcionamiento Global del CAE	3 días	mié 12/09/12	vie 14/09/12	59	EMM
64		Graficar pruebas de Constancia de la Calidad de la Imagen	3 días	lun 17/09/12	mié 19/09/12	63	EMM
65		Graficar pruebas de Evaluación de Impresoras	4 días	mié 19/09/12	lun 24/09/12	64	EMM
66		**Control de Acceso**	**2 días**	**lun 30/07/12**	**mar 31/07/12**	**31**	
67		Pantalla de control de acceso	1 día	lun 30/07/12	lun 30/07/12		EMM
68		Proceso de verificación de usuario	1 día	mar 31/07/12	mar 31/07/12	67	EMM
69		**Administración de usuarios**	**3 días**	**jue 02/08/12**	**lun 06/08/12**	**66**	
70		Pantalla de mantenimiento de usuarios	1 día	jue 02/08/12	jue 02/08/12		EMM
71		Asociar perfil al usuario	1 día	jue 02/08/12	jue 02/08/12	70	EMM
72		Asociar unidad al usuario	1 día	vie 03/08/12	vie 03/08/12	71	EMM
73		Pantalla de mantenimiento de perfiles	1 día	lun 06/08/12	lun 06/08/12		EMM

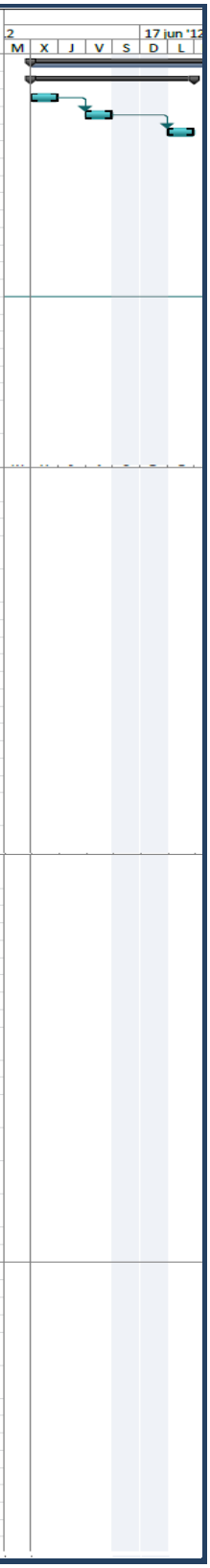


Figura 4.3 ***Cronograma basado de acuerdo a la solución propuesta.***

Actividad 3. Casos de uso de los módulos propuestos.

Los casos de uso permiten visualizar la secuencia del sistema desde el punto de vista del usuario y del desarrollador en diferentes momentos de su elaboración (antes, durante y después). Por medio de estos casos de uso se representamos la comunicación y el comportamiento de la solución de software mediante la interacción con los usuarios con el propio software.

A continuación mostramos el diagrama de caso de uso de los módulos de la solución de software.

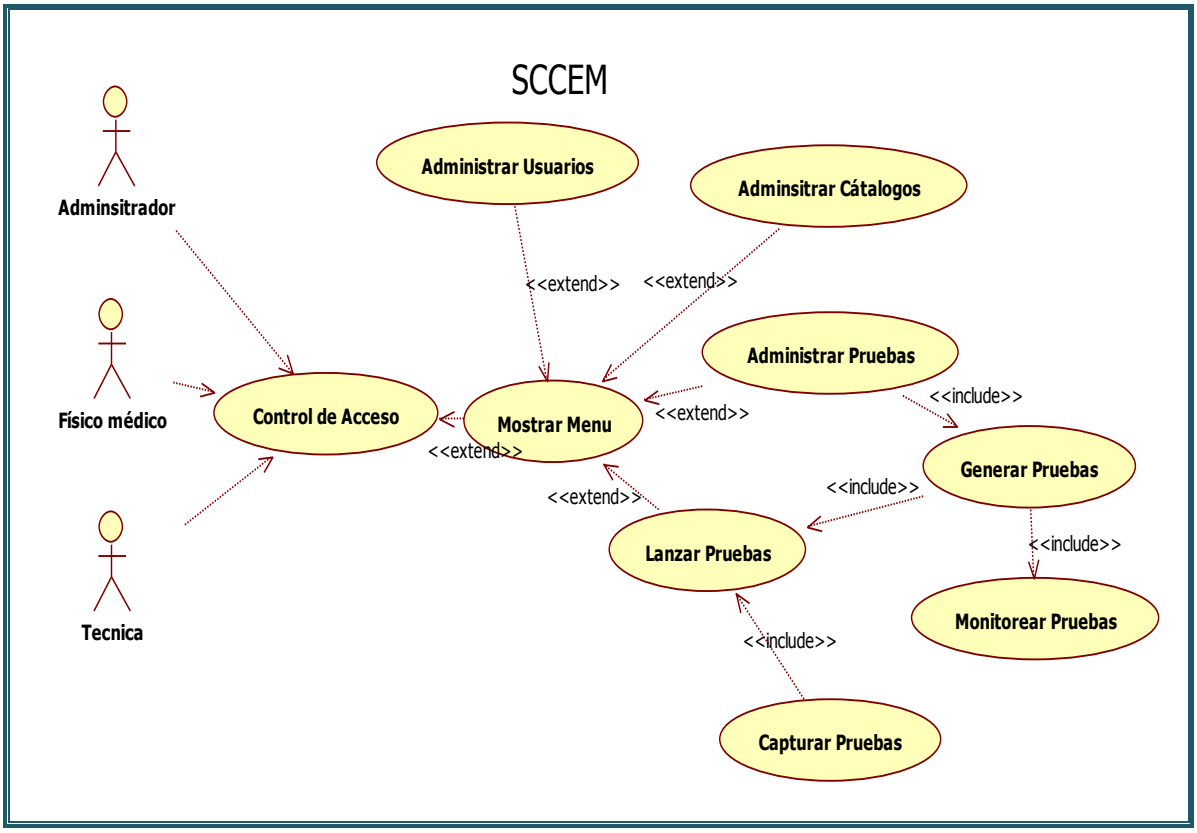


Figura 4.1 ***Casos de uso general de la solución propuesta.***

Con la narrativa de los **casos de uso** podemos resaltar los propósitos de las funciones o requerimientos.

Caso de uso: ***"Controlar Acceso"***

Descripción: Este caso de uso documenta el acceso de los usuarios al sistema.

Actores: Técnico Radiólogo, Físico Médico, Administrador, Sistema

Pre condiciones

1. El sistema verifica la autentificación del usuario en la base de datos.

Flujo principal

1. El sistema muestra las opciones del caso de uso "Mostar menú" al usuario.

Flujos alternos

1.1 El usuario decide salir del sistema.

Excepciones No hay.
Post condiciones No hay.

---

*Caso de uso: **"Mostrar Menú"***

Descripción: Este caso de uso documenta las opciones que el sistema muestra al usuario.

Actores: Técnico Radiólogo, Físico Médico, Administrador, Sistema

Pre condiciones

1. El sistema verifica la autentificación del usuario.

Flujo principal

1. El sistema muestra las opciones al usuario.
2. El usuario selecciona la opción deseada.
3. El sistema muestra la opción seleccionada.

Flujos alternos

2.1 El usuario decide salir del sistema.
2.2 Se ejecuta el caso de uso "Administración".
2.3 Se ejecuta el caso de uso "Registro de Pruebas".
2.4 Se ejecuta el caso de uso "Seguridad".

Excepciones No hay.
Post condiciones No hay.

---

*Caso de uso: **"Administrar Usuarios"***

Descripción: Este caso de uso documenta las opciones que el sistema muestra las opciones para administrar el registro del usuarios.

Actores: Físico Médico, Administrador, Sistema.

Pre condiciones

1. El sistema verifica la autentificación del usuario.

Flujo principal

1. El sistema muestra las opciones al usuario.
2. El usuario selecciona la opción deseada.
3. El sistema muestra la opción seleccionada.

Flujos alternos

2.1 El usuario decide salir del sistema.
2.2 Se ejecuta el caso de uso "Registrar Usuario".
2.3 Se ejecuta el caso de uso "Búsqueda de usuarios".

Excepciones No hay.
Post condiciones No hay.

Caso de uso: ***"Administrar Catálogos"***

Descripción: Este caso de uso documenta las opciones que el sistema muestra las opciones para administrar catálogos.

Actores: Físico Médico, Administrador, Sistema.

Pre condiciones

1. El sistema verifica la autentificación del usuario.
2. El usuario selecciona la opción de administrar catálogos.

Flujo principal

1. El sistema muestra las opciones al usuario.
2. El usuario selecciona la opción el catálogo deseado.
3. El sistema muestra la opción seleccionada.

Flujos alternos

2.1 El usuario decide salir del sistema.
2.2 Se ejecuta el caso de uso "Mantener catálogo de unidades".
2.3 Se ejecuta el caso de uso "Mantener catálogo de seguridad".
2.4 Se ejecuta el caso de uso "Mantener catálogo de instrumentos".
2.5 Se ejecuta el caso de uso "Mantener catálogo de tipo de mastógrafos".
2.6 Se ejecuta el caso de uso "Mantener catálogo de entidad federativa".
2.7 Se ejecuta el caso de uso "Mantener catálogo de frecuencia".
2.8 Se ejecuta el caso de uso "Mantener catálogo de pruebas".
2.9 Se ejecuta el caso de uso "Mantener catálogo de unidad de tiempo".

Excepciones No hay.
Post condiciones No hay.

Caso de uso: ***"Monitorear Pruebas"***

Descripción: Este caso de uso documenta el monitoreo de pruebas en el sistema.

Actores: Técnico Radiólogo**,** Físico Médico, Administrador, Sistema.

Pre condiciones

1. El sistema verifica la autentificación del usuario.
2. El usuario selecciona la opción de monitorear pruebas.
3. Se ejecuta el caso de uso "Generar Pruebas".
4. Se ejecuta el caso de uso "Administrar Pruebas".

Flujo principal

1. El sistema muestra un listado de las pruebas programadas para capturar.
2. El usuario selecciona la prueba que desea capturar.
3. El sistema muestra la opción seleccionada.
4. El usuario registra la información requerida.
5. El sistema guarda la información.

Flujos alternos

2.1 El usuario decide salir del sistema.
2.2 Se ejecuta el caso de uso "Mostrar Menú".
2.3 Se ejecuta el reporte correspondiente a la prueba seleccionada.
2.4 Se realiza el cambio de estatus de la prueba seleccionada.

Excepciones

1.1 Se muestra reporte de pruebas programadas.

Post condiciones No hay.

Caso de uso: ***"Administrar Pruebas"***

Descripción: Este caso de uso documenta la definición y asociación de las pruebas.

Actores: Físico Médico, Administrador, Sistema.

Pre condiciones

1. El sistema verifica la autentificación del usuario.

Flujo principal

1. El sistema muestra el listado de las pruebas definidas.
2. El usuario selecciona la prueba que desea para definir los parámetros.
3. El sistema muestra la opción seleccionada.
4. El usuario registra la definición de la prueba.
5. El sistema guarda la información.

Flujos alternos

2.1 El usuario decide salir del sistema.
2.2 El sistema regresa el listado inicial.

Excepciones No hay.

Post condiciones No hay.

Caso de uso: ***"Generar Pruebas"***

Descripción: Este caso de uso documenta la programación de pruebas.

Actores: Físico Médico, Administrador, Sistema.

Pre condiciones

1. El sistema verifica la autentificación del usuario.

Flujo principal

1. El sistema muestra el listado de las pruebas definidas en la unidad.
2. El usuario selecciona la prueba que desea para programar.
3. El sistema muestra la opción seleccionada.
4. El usuario realiza un comentario para comenzar la programación.
5. El sistema procesa y guarda la información.

Flujos alternos

2.1 El usuario decide salir del sistema.
2.2 El sistema regresa el listado inicial.

Excepciones

5.1 Muestra un mensaje de error en caso de no haber realizado el proceso de programación correctamente.

Post condiciones No hay.

Caso de uso: ***"Capturar Pruebas"***

Descripción: Este caso de uso documenta la definición y asociación de las pruebas.

Actores: Físico Médico, Administrador, Sistema.

Pre condiciones

1. El sistema verifica la autentificación del usuario.

Flujo principal

1. El sistema muestra el listado de las pruebas programadas de acuerdo a una tolerancia de captura.
2. El usuario selecciona la prueba que desea capturar.
3. El sistema muestra la opción seleccionada.
4. El usuario registra la prueba.
5. El sistema guarda la información.

Flujos alternos

2.1 El usuario decide salir del sistema.
2.2 El sistema regresa el listado inicial.

Excepciones No hay.
Post condiciones No hay.[5]

[5] Casos de uso de los módulos generales propuestos para la solución.

Con el análisis llegamos a tener una concepción de la solución de software y podemos comenzar con el diseño, desarrollo e implementación, de esta manera tenemos claro la definición de la solución.

El diagrama de casos de uso es una perspectiva general de la solución de software. Con este diagrama de casos de uso podemos ver la interacción e interdependencia entre cada uno de los casos de uso considerando la secuencia y comportamiento de los mismos.

El diagrama de casos de uso nos permite representar elementos que tienen asociada una funcionalidad común de una forma clara y sencilla, sin tener que demasiados formalismo, se lee de izquierda a derecha de arriba hacia abajo con sus respectivas relaciones que indican la obligatoriedad (<*include*>) y opcionalidad (<*extend*>), nos permite estimar tiempos, costos para cumplir con los requerimientos y el objetivo del proyecto.

2. DISEÑO

El diseño puede considerarse como la primera etapa para el desarrollo de cualquier solución de software y consta de diversos procesos, técnicas y principios con la intención de definir el proceso de una solución de software. El diseño produce diseño de interfaz, diseño de datos y mapa de navegación.

2.1 PROTOTIPO

Un prototipo es una representación limitada de la solución, que permite probar situaciones reales o explorar su uso, creando así un proceso de diseño iterativo. Un prototipo puede ser desde un simple dibujo o un software más complejo.

Con el prototipo se muestra parte del diseño de interfaz de la aplicación como el flujo de información y la descripción de algunos módulos o procesos.

Es frecuente que los usuarios no sepan lo que quieren, pero cuando ven algo y lo utilizan, pronto sugieren innovaciones para mejorar las ideas originales.

Esto ayuda a identificar, comunicar y probar la solución antes de crearla. Una vez aprobado del prototipo podemos comenzar el desarrollo.

A continuación se muestra la imagen de un prototipo de uno de los protocolos que nos solicitaron.

HOJA DE REGISTRO ANÁLISIS DE ESTUDIOS REPETIDOS

Hospital: _____ **Fecha:** / /

Mastógrafo					
Marca:		Modelo:		N/S:	

Fecha de inicio del periodo de evaluación: _____ Fecha de término del periodo de evaluación: _____

MARCAR EL RECUADRO QUE CORRESPONDA AL MOTIVO Y CANTIDAD DE PELÍCULAS REPETIDAS O RECHAZADAS

Motivo del rechazo	Cantidad de películas	Total:	%
1. Mala colocación de la paciente			
2. Movimiento de la paciente			
3. Radiografía muy clara			
4. Radiografía muy oscura			
5. Artefactos			
6. Velo			
7. Fallas en el revelado (No aplica para equipos CR y digitales)			
8. Información deficiente acerca de la paciente			
9. Otras razones			
10. Control de calidad (no se incluyen en el conteo)			
	Total de Rechazos (1- 10)		
	Total de repeticiones (1-9)		
		#	**%**
	Total de película consumida durante todo el periodo de evaluación:		100

REALIZÓ: _____

totalRechazadas = suma de las categorias de la 1 a la 10

totalRepetidas = totalRechazadas menos la categoria de control de calidad

porcentajeRechazadas = totalRechazadas /total de pelicula usada (consumida) * 100

porcentaje por categoria = total de la caterogria /totalRepetidas *100 (esto es por cada una de las categorias)

porcentajeRepetidas = totalRepetidas/total de pelicula usada (consumida) *100 y este porcentaje se toma para el criterio de aceptacion

Figura 4.1 ***Prototipo de un protocolo de basado en el MCCEM.***

Despues de la revisión con el usuario surgio la siguiente propuesta misma que aplicamos al resto del sistema.

REGISTRO DE ANÁLISIS DE ESTUDIOS REPETIDOS

Fecha de inicio del periodo de evaluación: 14/08/2012 Fecha de término del periodo de evaluación: 14/08/2012

MARCAR EL RECUADRO QUE CORRESPONDA AL MOTIVO Y CANTIDAD DE PELÍCULAS REPETIDAS O RECHAZADAS

MALA COLOCACIÓN	MOVIMIENTO DEL PACIENTE	IMAGEN MUY CLARA	IMAGEN MUY OSCURA	ARTEFACTOS	VELO	FALLAS EN EL REVELADO	INFORMACIÓN DEFICIENTE DEL PAC.	OTRAS RAZONES	CONTROL DE CALIDAD	TOTAL DE PELÍCULAS USADAS	TOTAL PELÍCULAS RECHAZADAS	TOTAL DE PELÍCULAS REPETIDAS	CONCLUSIÓN
2	2	7	1	0	1	0	0	4	0	500	17	17	
%	%	%	%	%	%	%	%	%		%	%	%	
11.76	11.76	41.18	5.88	0	5.88	0	0	23.53		100	3.4	3.4	

Figura 4.2 ***Prototipo propuesto.***

2.2 MODELADO DE DATOS

El modelado de datos es la manera lógica en que conceptualizamos el dominio de la información para nuestra solución de software. Revisamos y consensamos acuerdos con nuestro equipo de trabajo para definir la existencia de los datos en la base de datos, además de que se tomaron en cuenta las tres primeras formas normales del diseño para una base de datos.

Para nuestra solución fue importante la relación de cómo se definieron las pruebas y que elementos se evalúan en cada una de ellas de acuerdo al número y tipo de instrumentos existentes en cada unidad médica. Lo anterior es importante porque de esto depende el número de procesos, funciones y los tiempos que se requerirán para ingresar y mostrar la información. Lo que se busca es facilitar o hacer de forma más transparente el flujo de información en el sistema.

Para crear las tablas y los campos consultamos la información recopilada en el análisis, siendo lo más importante el considerar atributos como los tipos de datos, la longitud, nombre, reglas de derivación u obtención, rango de valores, formato, etc.

A continuación se muestra la parte más importante del diagrama Entidad-Relación, porque se definen los protocolos de acuerdo al número de elementos a evaluar en cada unidad médica y con esto podemos realizar la programación de los mismos.

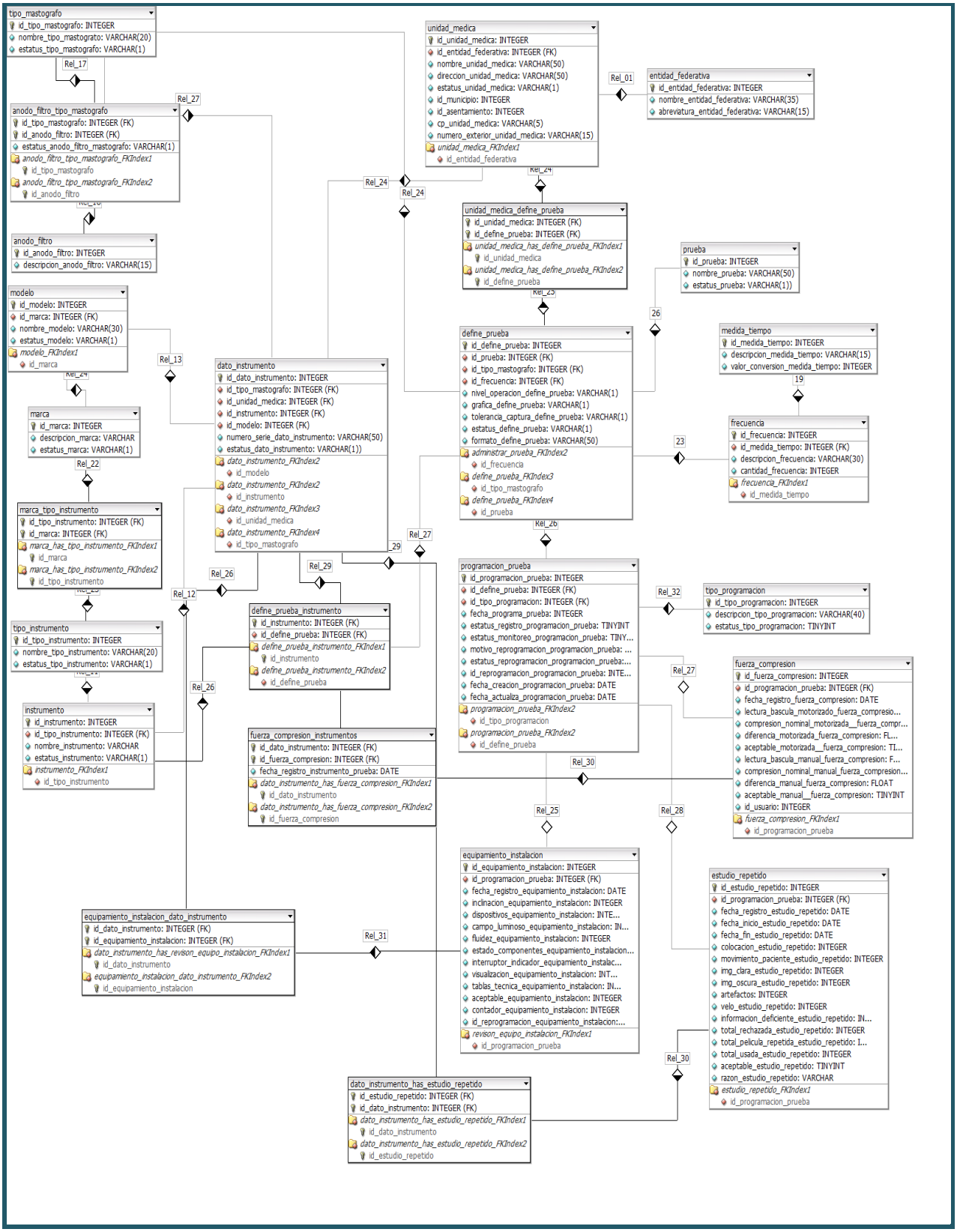


Figura 4.3 ***Esquema Lógico del Diagrama Entidad Relación.***

2.3 LISTADO DE FUNCIONES

El listado de las funciones ó programas consiste en un conjunto de tareas que se entrelazan para realizar y cumplir el propósito u objetivo de los requerimientos.

Las funciones las clasificamos de acuerdo a un tipo de programa como son procesos, pantalla o interfaz visual y reportes además de que cumplen con un propósito específico o por cómo está formado el proyecto, algunos son los archivos de configuración, interfaz y entidades de negocio entre otros.

La lista de funciones que se muestra a continuación es la final la cual se fue actualizando en paralelo con los avances del desarrollo. Las funciones importantes para el comienzo del proyecto y que dan estructura son los archivos de configuración.

No.	Función	Tipo de Programa	Nombre del Programa
1	Archivo de configuración	Proceso	criteria.properties
2	Archivo de configuración	Proceso	global.properties
3	Archivo de configuración	Proceso	hibernate.cfg.xml
4	Archivo de configuración	Proceso	log4j.properties
5	Archivo de configuración	Proceso	parameters.properties
6	Archivo de configuración	Proceso	struts.properties
7	Archivo de configuración	Proceso	struts.xml
8	Archivo de configuración	Proceso	struts2.xml
9	Archivo de configuración	Proceso	struts3.xml
10	Archivo de configuración	Proceso	struts4.xml
11	Archivo de configuración	Proceso	styles.css
12	Archivo de configuración	Proceso	theme.properties
13	Archivo de configuración	Proceso	validation.js
14	Interfaz	Proceso	AgendaDao.java
15	Interfaz	Proceso	AnodoFiltroDao.java
16	Interfaz	Proceso	ArtefactoCrDao.java
17	Interfaz	Proceso	ArtefactoCrReferenciaDao.java
18	Interfaz	Proceso	ArtefactoDrDao.java
19	Interfaz	Proceso	ArtefactoDrReferenciaDao.java
20	Interfaz	Proceso	CalidadImagenDao.java
21	Interfaz	Proceso	CalidadImagenReferenciaDao.java
22	Interfaz	Proceso	ConstanciaGlobalDao.java
23	Interfaz	Proceso	ConstanciaGlobalReferenciaDao.java
24	Interfaz	Proceso	CuartoOscuroDao.java
25	Interfaz	Proceso	DatoInstrumentoDao.java
26	Interfaz	Proceso	DefinePruebaDao.java
27	Interfaz	Proceso	DireccionDao.java
28	Interfaz	Proceso	EntidadFederativaDao.java
29	Interfaz	Proceso	EquipamientoInstalacionDao.java

30	Interfaz	Proceso	EstadoCivilDao.java
31	Interfaz	Proceso	EstudioRepetidoDao.java
32	Interfaz	Proceso	EvaluaImpresoraInspeccionDao.java
33	Interfaz	Proceso	EvaluaImpresoraNivelHomogeneidadDao.java
34	Interfaz	Proceso	EvaluaImpresoraSensitometriaDao.java
35	Interfaz	Proceso	EvaluaImpresoraSReferenciaDao.java
36	Interfaz	Proceso	EvaluaMonitorDao.java
37	Interfaz	Proceso	EvaluaMonitorInspeccionDao.java
38	Interfaz	Proceso	FrecuenciaDao.java
39	Interfaz	Proceso	FuerzaCompresionDao.java
40	Interfaz	Proceso	InstrumentoDao.java
41	Interfaz	Proceso	MapaDao.java
42	Interfaz	Proceso	MarcaDao.java
43	Interfaz	Proceso	MedidaTiempoDao.java
44	Interfaz	Proceso	ModeloDao.java
45	Interfaz	Proceso	ModoExposicionDao.java
46	Interfaz	Proceso	PeliculaPantallaDao.java
47	Interfaz	Proceso	PerfilDao.java
48	Interfaz	Proceso	PosicionDetectorDao.java
49	Interfaz	Proceso	PosicionSelectorDao.java
50	Interfaz	Proceso	ProcesadorPeliculaReferenciaDetalleDao.java
51	Interfaz	Proceso	ProgramacionPruebaDao.java
52	Interfaz	Proceso	PruebaDao.java
53	Interfaz	Proceso	ReceptorImagenDao.java
54	Interfaz	Proceso	RegistroCalidadImagenActualDao.java
55	Interfaz	Proceso	RegistroCalidadImagenDao.java
56	Interfaz	Proceso	RetencionFijadorDao.java
57	Interfaz	Proceso	SeccionDao.java
58	Interfaz	Proceso	SexoDao.java
59	Interfaz	Proceso	TipoInstrumentoDao.java
60	Interfaz	Proceso	TipoMastografoDao.java
61	Interfaz	Proceso	TipoProgramacionDao.java
62	Interfaz	Proceso	UnidadMedicaDao.java
63	Interfaz	Proceso	UsuarioDao.java
64	Implementación de la Interfaz	Proceso	AgendaDaoImp.java
65	Implementación de la Interfaz	Proceso	AnodoFiltroDaoImp.java
66	Implementación de la Interfaz	Proceso	ArtefactoCrDaoImp.java
67	Implementación de la Interfaz	Proceso	ArtefactoCrReferenciaDaoImp.java
68	Implementación de la Interfaz	Proceso	ArtefactoDrDaoImp.java
69	Implementación de la Interfaz	Proceso	ArtefactoDrReferenciaDaoImp.java
70	Implementación de la Interfaz	Proceso	CalidadImagenDaoImp.java
71	Implementación de la Interfaz	Proceso	CalidadImagenReferenciaDaoImp.java
72	Implementación de la Interfaz	Proceso	ConstanciaGlobalDaoImp.java
73	Implementación de la Interfaz	Proceso	ConstanciaGlobalReferenciaDaoImp.java
74	Implementación de la Interfaz	Proceso	CuartoOscuroDaoImp.java
75	Implementación de la Interfaz	Proceso	DatoInstrumentoDaoImp.java
76	Implementación de la Interfaz	Proceso	DefinePruebaDaoImp.java

77	Implementación de la Interfaz	Proceso	DireccionDaoImpl.java
78	Implementación de la Interfaz	Proceso	EntidadFederativaDaoImp.java
79	Implementación de la Interfaz	Proceso	EquipamientoInstalacionDaoImp.java
80	Implementación de la Interfaz	Proceso	EstadoCivilDaoImpl.java
81	Implementación de la Interfaz	Proceso	EstudioRepetidoDaoImp.java
82	Implementación de la Interfaz	Proceso	EvaluaImpresoraInspeccionDaoImp.java
83	Implementación de la Interfaz	Proceso	EvaluaImpresoraNivelHomogeneidadDaoImp.java
84	Implementación de la Interfaz	Proceso	EvaluaImpresoraSensitometriaDaoImp.java
85	Implementación de la Interfaz	Proceso	EvaluaImpresoraSReferenciaDaoImp.java
86	Implementación de la Interfaz	Proceso	EvaluaMonitorDaoImp.java
87	Implementación de la Interfaz	Proceso	EvaluaMonitorInspeccionDaoImp.java
88	Implementación de la Interfaz	Proceso	FrecuenciaDaoImp.java
89	Implementación de la Interfaz	Proceso	FuerzaCompresionDaoImp.java
90	Implementación de la Interfaz	Proceso	InstrumentoDaoImp.java
91	Implementación de la Interfaz	Proceso	MapaDaoImp.java
92	Implementación de la Interfaz	Proceso	MarcaDaoImp.java
93	Implementación de la Interfaz	Proceso	MedidaTiempoDaoImp.java
94	Implementación de la Interfaz	Proceso	ModeloDaoImp.java
95	Implementación de la Interfaz	Proceso	ModoExposicionDaoImp.java
96	Implementación de la Interfaz	Proceso	PeliculaPantallaDaoImp.java
97	Implementación de la Interfaz	Proceso	PerfilDaoImpl.java
98	Implementación de la Interfaz	Proceso	PosicionDetectorDaoImp.java
99	Implementación de la Interfaz	Proceso	PosicionSelectorDaoImp.java
100	Implementación de la Interfaz	Proceso	ProcesadorPeliculaReferenciaDetalleDaoImp.java
101	Implementación de la Interfaz	Proceso	ProgramacionPruebaDaoImp.java
102	Implementación de la Interfaz	Proceso	PruebaDaoImp.java
103	Implementación de la Interfaz	Proceso	ReceptorImagenDaoImp.java
104	Implementación de la Interfaz	Proceso	RegistroCalidadImagenActualDaoImp.java
105	Implementación de la Interfaz	Proceso	RegistroCalidadImagenDaoImp.java
106	Implementación de la Interfaz	Proceso	RetencionFijadorDaoImp.java
107	Implementación de la Interfaz	Proceso	SeccionDaoImpl.java
108	Implementación de la Interfaz	Proceso	SexoDaoImpl.java
109	Implementación de la Interfaz	Proceso	TipoInstrumentoDaoImp.java
110	Implementación de la Interfaz	Proceso	TipoMastografoDaoImp.java
111	Implementación de la Interfaz	Proceso	TipoProgramacionDaoImp.java
112	Implementación de la Interfaz	Proceso	UnidadMedicaDaoImp.java
113	Implementación de la Interfaz	Proceso	UsuarioDaoImp.java
114	Entidades de Negocio	Proceso	Agenda.java
115	Entidades de Negocio	Proceso	AnodoFiltro.java
116	Entidades de Negocio	Proceso	AnodoFiltroTipoMastografo.java
117	Entidades de Negocio	Proceso	AnodoFiltroTipoMastografoId.java
118	Entidades de Negocio	Proceso	ArtefactoCr.java
119	Entidades de Negocio	Proceso	ArtefactoCrReferencia.java
120	Entidades de Negocio	Proceso	ArtefactoCrReferenciaInstrumento.java
121	Entidades de Negocio	Proceso	ArtefactoCrReferenciaInstrumentoId.java
122	Entidades de Negocio	Proceso	ArtefactoDr.java
123	Entidades de Negocio	Proceso	ArtefactoDrReferencia.java

124	Entidades de Negocio	Proceso	ArtefactoDrReferenciaInstrumento.java
125	Entidades de Negocio	Proceso	ArtefactoDrReferenciaInstrumentoId.java
126	Entidades de Negocio	Proceso	Asentamiento.java
127	Entidades de Negocio	Proceso	AsentamientoId.java
128	Entidades de Negocio	Proceso	CalidadImagen.java
129	Entidades de Negocio	Proceso	CalidadImagenGrafica.java
130	Entidades de Negocio	Proceso	CalidadImagenReferencia.java
131	Entidades de Negocio	Proceso	CalidadImagenReferenciaInstrumento.java
132	Entidades de Negocio	Proceso	CalidadImagenReferenciaInstrumentoId.java
133	Entidades de Negocio	Proceso	ConstanciaGlobal.java
134	Entidades de Negocio	Proceso	ConstanciaGlobalGrafica.java
135	Entidades de Negocio	Proceso	ConstanciaGlobalReferencia.java
136	Entidades de Negocio	Proceso	ConstanciaGlobalReferenciaInstrumento.java
137	Entidades de Negocio	Proceso	ConstanciaGlobalReferenciaInstrumentoId.java
138	Entidades de Negocio	Proceso	CuartoOscuro.java
139	Entidades de Negocio	Proceso	CuartoOscuroInstrumento.java
140	Entidades de Negocio	Proceso	CuartoOscuroInstrumentoId.java
141	Entidades de Negocio	Proceso	DatoInstrumento.java
142	Entidades de Negocio	Proceso	DatosGrafica.java
143	Entidades de Negocio	Proceso	DefinePrueba.java
144	Entidades de Negocio	Proceso	DefinePruebaInstrumento.java
145	Entidades de Negocio	Proceso	DefinePruebaInstrumentoId.java
146	Entidades de Negocio	Proceso	DefinePruebaNombreLargo.java
147	Entidades de Negocio	Proceso	DiaSemana.java
148	Entidades de Negocio	Proceso	Direccion.java
149	Entidades de Negocio	Proceso	Entidad.java
150	Entidades de Negocio	Proceso	EntidadFederativa.java
151	Entidades de Negocio	Proceso	EquipamientoInstalacion.java
152	Entidades de Negocio	Proceso	EquipamientoInstalacionInstrumento.java
153	Entidades de Negocio	Proceso	EquipamientoInstalacionInstrumentoId.java
154	Entidades de Negocio	Proceso	EstadoCivil.java
155	Entidades de Negocio	Proceso	EstudioRepetido.java
156	Entidades de Negocio	Proceso	EstudioRepetidoInstrumento.java
157	Entidades de Negocio	Proceso	EstudioRepetidoInstrumentoId.java
158	Entidades de Negocio	Proceso	EvaluaImpresoraInspeccion.java
159	Entidades de Negocio	Proceso	EvaluaImpresoraInspeccionInstrumento.java
160	Entidades de Negocio	Proceso	EvaluaImpresoraInspeccionInstrumentoId.java
161	Entidades de Negocio	Proceso	EvaluaImpresoraNivelHomogeneidad.java
162	Entidades de Negocio	Proceso	EvaluaImpresoraNivelHomogeneidadInstrumento.java
163	Entidades de Negocio	Proceso	EvaluaImpresoraNivelHomogeneidadInstrumentoId.java
164	Entidades de Negocio	Proceso	EvaluaImpresoraSensitometria.java
165	Entidades de Negocio	Proceso	EvaluaImpresoraSensitometriaGrafica.java
166	Entidades de Negocio	Proceso	EvaluaImpresoraSensitometriaInstrumento.java
167	Entidades de Negocio	Proceso	EvaluaImpresoraSensitometriaInstrumentoId.java
168	Entidades de Negocio	Proceso	EvaluaImpresoraSReferencia.java
169	Entidades de Negocio	Proceso	EvaluaImpresoraSReferenciaInstrumento.java
170	Entidades de Negocio	Proceso	EvaluaImpresoraSReferenciaInstrumentoId.java

171	Entidades de Negocio	Proceso	EvaluaMonitor.java
172	Entidades de Negocio	Proceso	EvaluaMonitorInspeccion.java
173	Entidades de Negocio	Proceso	EvaluaMonitorInspeccionInstrumento.java
174	Entidades de Negocio	Proceso	EvaluaMonitorInspeccionInstrumentoId.java
175	Entidades de Negocio	Proceso	EvaluaMonitorInstrumento.java
176	Entidades de Negocio	Proceso	EvaluaMonitorInstrumentoId.java
177	Entidades de Negocio	Proceso	Frecuencia.java
178	Entidades de Negocio	Proceso	FuerzaCompresion.java
179	Entidades de Negocio	Proceso	FuerzaCompresionInstrumento.java
180	Entidades de Negocio	Proceso	FuerzaCompresionInstrumentoId.java
181	Entidades de Negocio	Proceso	Instrumento.java
182	Entidades de Negocio	Proceso	Mapa.java
183	Entidades de Negocio	Proceso	Marca.java
184	Entidades de Negocio	Proceso	MarcaTipoInstrumento.java
185	Entidades de Negocio	Proceso	MarcaTipoInstrumentoId.java
186	Entidades de Negocio	Proceso	MedidaTiempo.java
187	Entidades de Negocio	Proceso	Mes.java
188	Entidades de Negocio	Proceso	Modelo.java
189	Entidades de Negocio	Proceso	ModoExposicion.java
190	Entidades de Negocio	Proceso	Municipio.java
191	Entidades de Negocio	Proceso	MunicipioId.java
192	Entidades de Negocio	Proceso	PeliculaPantalla.java
193	Entidades de Negocio	Proceso	PeliculaPantallaChasis.java
194	Entidades de Negocio	Proceso	PeliculaPantallaChasisId.java
195	Entidades de Negocio	Proceso	PeliculaPantallaInstrumento.java
196	Entidades de Negocio	Proceso	PeliculaPantallaInstrumentoId.java
197	Entidades de Negocio	Proceso	Perfil.java
198	Entidades de Negocio	Proceso	PerfilSeccion.java
199	Entidades de Negocio	Proceso	PerfilSeccionId.java
200	Entidades de Negocio	Proceso	PosicionDetector.java
201	Entidades de Negocio	Proceso	PosicionSelector.java
202	Entidades de Negocio	Proceso	ProcesadorPelicula.java
203	Entidades de Negocio	Proceso	ProcesadorPeliculaReferenciaDetalle.java
204	Entidades de Negocio	Proceso	ProcesadorPeliculaReferenciaInstrumento.java
205	Entidades de Negocio	Proceso	ProcesadorPeliculaReferenciaInstrumentoId.java
206	Entidades de Negocio	Proceso	ProgramacionPrueba.java
207	Entidades de Negocio	Proceso	Prueba.java
208	Entidades de Negocio	Proceso	ReceptorImagen.java
209	Entidades de Negocio	Proceso	ReceptorImagenInstrumento.java
210	Entidades de Negocio	Proceso	ReceptorImagenInstrumentoId.java
211	Entidades de Negocio	Proceso	RegistroCalidadImagen.java
212	Entidades de Negocio	Proceso	RegistroCalidadImagenActual.java
213	Entidades de Negocio	Proceso	RegistroCalidadImagenActualGrafica.java
214	Entidades de Negocio	Proceso	RegistroCalidadImagenInstrumento.java
215	Entidades de Negocio	Proceso	RegistroCalidadImagenInstrumentoId.java
216	Entidades de Negocio	Proceso	RetencionFijador.java
217	Entidades de Negocio	Proceso	RetencionFijadorInstrumento.java

218	Entidades de Negocio	Proceso	RetencionFijadorInstrumentoId.java
219	Entidades de Negocio	Proceso	Seccion.java
220	Entidades de Negocio	Proceso	Semana.java
221	Entidades de Negocio	Proceso	Sexo.java
222	Entidades de Negocio	Proceso	TipoInstrumento.java
223	Entidades de Negocio	Proceso	TipoMastografo.java
224	Entidades de Negocio	Proceso	TipoProgramacion.java
225	Entidades de Negocio	Proceso	UnidadMedica.java
226	Entidades de Negocio	Proceso	UnidadMedicaDefinePrueba.java
227	Entidades de Negocio	Proceso	UnidadMedicaDefinePruebaId.java
228	Entidades de Negocio	Proceso	UnidadMedicaInstrumento.java
229	Entidades de Negocio	Proceso	Usuario.java
230	Entidades de Negocio	Proceso	UsuarioPerfil.java
231	Entidades de Negocio	Proceso	UsuarioPerfilId.java
232	Controlador	Proceso	GraficaAction.java
233	Controlador	Proceso	GraficaConstanciaGlobalAction.java
234	Controlador	Proceso	GraficaEvaluaImpresoraSensitometriaAction.java
235	Controlador	Proceso	GraficaRegistroCalidadImagenAction.java
236	Controlador	Proceso	LanzamientoGraphicAction.java
237	Controlador	Proceso	ValidateInterceptors.java
238	Controlador	Proceso	ArtefactoCrReporte.java
239	Controlador	Proceso	ArtefactoDrReporte.java
240	Controlador	Proceso	CalidadImagenReporte.java
241	Controlador	Proceso	ConstanciaGlobalReporte.java
242	Controlador	Proceso	CuartoOscuroReporte.java
243	Controlador	Proceso	EquipamientoInstalacionReporte.java
244	Controlador	Proceso	EstudioRepetidoReporte.java
245	Controlador	Proceso	EvaluaImpresoraInspeccionReporte.java
246	Controlador	Proceso	EvaluaImpresoraNivelHomogeneidadReporte.java
247	Controlador	Proceso	EvaluaImpresoraSensitometriaReporte.java
248	Controlador	Proceso	EvaluaMonitorInspeccionReporte.java
249	Controlador	Proceso	EvaluaMonitorReporte.java
250	Controlador	Proceso	FuerzaCompresionReporte.java
251	Controlador	Proceso	PeliculaPantallaReporte.java
252	Controlador	Proceso	ProgramacionResumen.java
253	Controlador	Proceso	PruebaReporte.java
254	Controlador	Proceso	ReceptorImagenReporte.java
255	Controlador	Proceso	RegistroCalidadImagenActualReporte.java
256	Controlador	Proceso	RetencionFijadorReporte.java
257	Controlador	Proceso	AceptabilidadService.java
258	Controlador	Proceso	ArtefactoCrService.java
259	Controlador	Proceso	ArtefactoDrService.java
260	Controlador	Proceso	Calendario.java
261	Controlador	Proceso	CalidadImagenService.java
262	Controlador	Proceso	ConstanciaGlobalService.java
263	Controlador	Proceso	CuartoOscuroService.java
264	Controlador	Proceso	EntidadService.java

265	Controlador	Proceso	EquipamientoInstalacionService.java
266	Controlador	Proceso	EstudioRepetidoService.java
267	Controlador	Proceso	EvaluaImpresoraInspeccionService.java
268	Controlador	Proceso	EvaluaImpresoraNivelHomogeneidadService.java
269	Controlador	Proceso	EvaluaMonitorInspeccionService.java
270	Controlador	Proceso	EvaluaMonitorService.java
271	Controlador	Proceso	FuerzaCompresionService.java
272	Controlador	Proceso	Mastografo.java
273	Controlador	Proceso	PeliculaPantallaService.java
274	Controlador	Proceso	ProcesadorPeliculaService.java
275	Controlador	Proceso	ProgramacionBusquedaService.java
276	Controlador	Proceso	ProgramacionService.java
277	Controlador	Proceso	ReceptorImagenService.java
278	Controlador	Proceso	RegistroCalidadImagenActualService.java
279	Controlador	Proceso	ResumenService.java
280	Controlador	Proceso	RetencionFijadorService.java
281	Controlador	Proceso	Session.java
282	Controlador	Proceso	DeleteImage.java
283	Controlador	Proceso	Graphic.java
284	Controlador	Proceso	GraphicXY.java
285	Controlador	Proceso	HibernateUtil.java
286	Controlador	Proceso	MyServletContextListener.java
287	Controlador	Proceso	ReziseImage.java
288	Utilería	Proceso	Utilities.java
289	Controlador	Proceso	AnodoFiltroAction.java
290	Controlador	Proceso	ArtefactoCrAction.java
291	Controlador	Proceso	ArtefactoCrReferenciaAction.java
292	Controlador	Proceso	ArtefactoDrAction.java
293	Controlador	Proceso	ArtefactoDrReferenciaAction.java
294	Controlador	Proceso	CalidadImagenAction.java
295	Controlador	Proceso	CalidadImagenReferenciaAction.java
296	Controlador	Proceso	ComboDependienteAction.java
297	Controlador	Proceso	ConstanciaGlobalAction.java
298	Controlador	Proceso	ConstanciaGlobalReferenciaAction.java
299	Controlador	Proceso	CuartoOscuroAction.java
300	Controlador	Proceso	DatoInstrumentoAction.java
301	Controlador	Proceso	DefinePruebaAction.java
302	Controlador	Proceso	DireccionAction.java
303	Controlador	Proceso	EntidadFederativaAction.java
304	Controlador	Proceso	EquipamientoInstalacionAction.java
305	Controlador	Proceso	EstudioRepetidoAction.java
306	Controlador	Proceso	EvaluacionSensitometriaAction.java
307	Controlador	Proceso	EvaluaImpresoraInspeccionAction.java
308	Controlador	Proceso	EvaluaImpresoraNivelHomogeneidadAction.java
309	Controlador	Proceso	EvaluaImpresoraSReferenciaAction.java
310	Controlador	Proceso	EvaluaMonitorAction.java
311	Controlador	Proceso	EvaluaMonitorInspeccionAction.java

312	Controlador	Proceso	FrecuenciaAction.java
313	Controlador	Proceso	FuerzaCompresionAction.java
314	Controlador	Proceso	GeneraInstrumentosAction.java
315	Controlador	Proceso	InformacionAction.java
316	Controlador	Proceso	InstrumentoAction.java
317	Controlador	Proceso	ListasAction.java
318	Controlador	Proceso	LoginAction.java
319	Controlador	Proceso	MapaAction.java
320	Controlador	Proceso	MarcasAction.java
321	Controlador	Proceso	MedidaTiempoAction.java
322	Controlador	Proceso	MenuAction.java
323	Controlador	Proceso	MessagesAction.java
324	Controlador	Proceso	ModeloAction.java
325	Controlador	Proceso	ModoExposicionAction.java
326	Controlador	Proceso	MonitoreoAction.java
327	Controlador	Proceso	MonitoreoxEstatusAction.java
328	Controlador	Proceso	PeliculaPantallaAction.java
329	Controlador	Proceso	PerfilAction.java
330	Controlador	Proceso	PosicionDetectorAction.java
331	Controlador	Proceso	PosicionSelectorAction.java
332	Controlador	Proceso	ProcesadorPeliculaReferenciaAction.java
333	Controlador	Proceso	ProgramacionAction.java
334	Controlador	Proceso	ProgramacionCapturaAction.java
335	Controlador	Proceso	PruebaAction.java
336	Controlador	Proceso	ReceptorImagenAction.java
337	Controlador	Proceso	RegistroCalidadImagenAction.java
338	Controlador	Proceso	RegistroCalidadImagenActual1Action.java
339	Controlador	Proceso	ArtefactoCrReporteAction.java
340	Controlador	Proceso	ArtefactoDrReporteAction.java
341	Controlador	Proceso	CalidadImagenReporteAction.java
342	Controlador	Proceso	ConstanciaGlobalReporteAction.java
343	Controlador	Proceso	CuartoOscuroReporteAction.java
344	Controlador	Proceso	EquipamientoInstalacionReporteAction.java
345	Controlador	Proceso	EstudioRepetidoReporteAction.java
346	Controlador	Proceso	EvaluaImpresoraInspeccionReporteAction.java
347	Controlador	Proceso	EvaluaImpresoraNivelHomogeneidadReporteAction.java
348	Controlador	Proceso	EvaluaImpresoraSensitometriaReporteAction.java
349	Controlador	Proceso	EvaluaMonitorInspeccionReporteAction.java
350	Controlador	Proceso	EvaluaMonitorReporteAction.java
351	Controlador	Proceso	FuerzaCompresionReporteAction.java
352	Controlador	Proceso	LanzamientoReporteAction.java
353	Controlador	Proceso	PeliculaPantallaReporteAction.java
354	Controlador	Proceso	ReceptorImagenReporteAction.java
355	Controlador	Proceso	RegistroCalidadImagenActualReporteAction.java
356	Controlador	Proceso	ResumenReporteAction.java
357	Controlador	Proceso	RetencionFijadorReporteAction.java
358	Controlador	Proceso	RetencionFijadorAction.java

359	Controlador	Proceso	SeccionAction.java
360	Controlador	Proceso	TipoInstrumentoAction.java
361	Controlador	Proceso	TipoMastografoAction.java
362	Controlador	Proceso	TipoProgramacionAction.java
363	Controlador	Proceso	UnidadMedicaAction.java
364	Controlador	Proceso	UnidadMedicaDefinePruebaAction.java
365	Controlador	Proceso	UsersAction.java
366	Controlador	Proceso	UsuarioAction.java
367	Controlador	Proceso	UsuarioRegistroAction.java
368	Controlador	Proceso	UsuarioSessionDatosAction.java
369	Controlador	Proceso	UtilerіasAction.java
370	Acceso	Pantalla	index.html
371	Informe	Reporte	artefactoCr.jasper
372	Informe	Reporte	artefactoDr.jasper
373	Informe	Reporte	calidadImages.jasper
374	Informe	Reporte	constanciaGlobal.jasper
375	Informe	Reporte	cuartoOscuro.jasper
376	Informe	Reporte	equipamientoInstalacion.jasper
377	Informe	Reporte	estudiosRepetidos.jasper
378	Informe	Reporte	evaluaMonitor.jasper
379	Informe	Reporte	evaluaMonitorInspeccion.jasper
380	Informe	Reporte	fuerzaCompresion.jasper
381	Informe	Reporte	impresorasImpresion.jasper
382	Informe	Reporte	impresorasNivelHomogeneidad.jasper
383	Informe	Reporte	impresorasSensitometria.jasper
384	Informe	Reporte	peliculaPantalla.jasper
385	Informe	Reporte	receptorImages.jasper
386	Informe	Reporte	registroCalidadImagenActual.jasper
387	Informe	Reporte	registroCalidadImagenActualSub.jasper
388	Informe	Reporte	resumenControl.jasper
389	Utilería	Reporte	retencionFijador.jasper
390	Seguridad	Pantalla	bus_usuarios.js
391	Acceso	Pantalla	menu.js
392	Utilería	Pantalla	titulos.js
393	Acceso	Pantalla	error500.jsp
394	Acceso	Pantalla	login.jsp
395	Acceso	Pantalla	mensaje.jsp
396	Catálogo	Pantalla	combo_asentamiento.jsp
397	Catálogo	Pantalla	combo_entidad.jsp
398	Catálogo	Pantalla	combo_municipio.jsp
399	Catálogo	Pantalla	combo_perfil.jsp
400	Catálogo	Pantalla	anodo_filtro.jsp
401	Catálogo	Pantalla	depending_modelo.jsp
402	Catálogo	Pantalla	depending_unidad_medica.jsp
403	Catálogo	Pantalla	instrumento.jsp
404	Catálogo	Pantalla	marcas.jsp
405	Catálogo	Pantalla	modo_exposicion.jsp

406	Catálogo	Pantalla	posicion_detector.jsp
407	Catálogo	Pantalla	posicion_selector.jsp
408	Catálogo	Pantalla	tabla_define_prueba.jsp
409	Catálogo	Pantalla	tabla_programacion.jsp
410	Catálogo	Pantalla	tabla_unidad_medica_define_prueba.jsp
411	Catálogo	Pantalla	ingresa_define_prueba.jsp
412	Catálogo	Pantalla	ingresa_programacion.jsp
413	Catálogo	Pantalla	ingresa_unidad_medica_define_prueba.jsp
414	Catálogo	Pantalla	main_define_prueba.jsp
415	Catálogo	Pantalla	main_programacion.jsp
416	Catálogo	Pantalla	main_unidad_medica_define_prueba.jsp
417	Catálogo	Pantalla	tabla_monitoreo.jsp
418	Catálogo	Pantalla	catalogo_anodo_filtro.jsp
419	Catálogo	Pantalla	catalogo_dato_instrumento.jsp
420	Catálogo	Pantalla	catalogo_marca.jsp
421	Catálogo	Pantalla	catalogo_modelo.jsp
422	Catálogo	Pantalla	catalogo_modo_exposicion.jsp
423	Catálogo	Pantalla	catalogo_posicion_detector.jsp
424	Catálogo	Pantalla	catalogo_posicion_selector.jsp
425	Catálogo	Pantalla	catalogo_tipo_mastografo.jsp
426	Catálogo	Pantalla	catalogo_tipo_programacion.jsp
427	Catálogo	Pantalla	tabla_anodo_filtro.jsp
428	Catálogo	Pantalla	tabla_dato_instrumento.jsp
429	Catálogo	Pantalla	tabla_entidad_federativa.jsp
430	Catálogo	Pantalla	tabla_frecuencia.jsp
431	Catálogo	Pantalla	tabla_instrumento.jsp
432	Catálogo	Pantalla	tabla_marca.jsp
433	Catálogo	Pantalla	tabla_medida_tiempo.jsp
434	Catálogo	Pantalla	tabla_modelo.jsp
435	Catálogo	Pantalla	tabla_modo_exposicion.jsp
436	Catálogo	Pantalla	tabla_posicion_detector.jsp
437	Catálogo	Pantalla	tabla_posicion_selector.jsp
438	Catálogo	Pantalla	tabla_prueba.jsp
439	Catálogo	Pantalla	tabla_tipo_instrumento.jsp
440	Catálogo	Pantalla	tabla_tipo_mastografo.jsp
441	Catálogo	Pantalla	tabla_tipo_programacion.jsp
442	Catálogo	Pantalla	tabla_unidad_medica.jsp
443	Catálogo	Pantalla	ingresa_anodo_filtro.jsp
444	Catálogo	Pantalla	ingresa_dato_instrumento.jsp
445	Catálogo	Pantalla	ingresa_entidad_federativa.jsp
446	Catálogo	Pantalla	ingresa_frecuencia.jsp
447	Catálogo	Pantalla	ingresa_instrumento.jsp
448	Catálogo	Pantalla	ingresa_marca.jsp
449	Catálogo	Pantalla	ingresa_medida_tiempo.jsp
450	Catálogo	Pantalla	ingresa_modelo.jsp
451	Catálogo	Pantalla	ingresa_modo_exposicion.jsp
452	Catálogo	Pantalla	ingresa_posicion_detector.jsp

453	Catálogo	Pantalla	ingresa_posicion_selector.jsp
454	Catálogo	Pantalla	ingresa_prueba.jsp
455	Catálogo	Pantalla	ingresa_tipo_instrumento.jsp
456	Catálogo	Pantalla	ingresa_tipo_mastografo.jsp
457	Catálogo	Pantalla	ingresa_tipo_programacion.jsp
458	Catálogo	Pantalla	ingresa_unidad_medica.jsp
459	Catálogo	Pantalla	main_entidad_federativa.jsp
460	Catálogo	Pantalla	main_frecuencia.jsp
461	Catálogo	Pantalla	main_instrumento.jsp
462	Catálogo	Pantalla	main_medida_tiempo.jsp
463	Catálogo	Pantalla	main_pruebas.jsp
464	Catálogo	Pantalla	main_tipo_instrumento.jsp
465	Catálogo	Pantalla	main_unidad_medica.jsp
466	Mapa	Pantalla	mapaDatos.jsp
467	Mapa	Pantalla	mapamex.jsp
468	Acceso	Pantalla	menu_arbol.jsp
469	Acceso	Pantalla	menu.jsp
470	Acceso	Pantalla	menu2.jsp
471	Monitoreo	Pantalla	tabla_monitoreo.jsp
472	Monitoreo	Pantalla	tabla_monitoreo_estatus.jsp
473	Monitoreo	Pantalla	tabla_monitoreo_programadas.jsp
474	Monitoreo	Pantalla	tabla_prueba.jsp
475	Monitoreo	Pantalla	main_monitoreo.jsp
476	Monitoreo	Pantalla	main_monitoreo_estatus.jsp
477	Seguridad	Pantalla	aso_perfiles_secciones.jsp
478	Seguridad	Pantalla	chk_secciones.jsp
479	Seguridad	Pantalla	ctg_perfiles.jsp
480	Seguridad	Pantalla	ing_perfiles.jsp
481	Seguridad	Pantalla	mnt_perfiles.jsp
482	Seguridad	Pantalla	ctg_secciones.jsp
483	Seguridad	Pantalla	ctg_secciones_arbol.jsp
484	Seguridad	Pantalla	ing_secciones.jsp
485	Seguridad	Pantalla	mnt_secciones.jsp
486	Seguridad	Pantalla	pdf_criterios.jsp
487	Seguridad	Pantalla	pdf_impresion.jsp
488	Catálogo	Pantalla	tabla_chasis.jsp
489	Catálogo	Pantalla	tabla_cuarto_oscuro.jsp
490	Control de Hojas de Registro	Pantalla	tabla_pruebas_programadas.jsp
491	Control de Hojas de Registro	Pantalla	main_pruebas_programadas.jsp
492	Registro de Hojas de Control	Pantalla	prueba_analisis_estudios_repetidos.jsp
493	Registro de Hojas de Control	Pantalla	prueba_analisis_retencion_fijador.jsp
494	Registro de Hojas de Control	Pantalla	prueba_artefactos_cr.jsp
495	Registro de Hojas de Control	Pantalla	prueba_artefactos_cr_referencia.jsp
496	Registro de Hojas de Control	Pantalla	prueba_artefactos_cr_referencia_consulta.jsp
497	Registro de Hojas de Control	Pantalla	prueba_artefactos_dr.jsp
498	Registro de Hojas de Control	Pantalla	prueba_artefactos_dr_referencia.jsp
499	Registro de Hojas de Control	Pantalla	prueba_artefactos_dr_referencia_consulta.jsp

500	Registro de Hojas de Control	Pantalla	prueba_calidad_imagen.jsp
501	Registro de Hojas de Control	Pantalla	prueba_calidad_imagen_referencia.jsp
502	Registro de Hojas de Control	Pantalla	prueba_calidad_imagen_referencia_consulta.jsp
503	Registro de Hojas de Control	Pantalla	prueba_constancia_global.jsp
504	Registro de Hojas de Control	Pantalla	prueba_constancia_global_referencia.jsp
505	Registro de Hojas de Control	Pantalla	prueba_constancia_global_referencia_consulta.jsp
506	Registro de Hojas de Control	Pantalla	prueba_cuarto_oscuro.jsp
507	Registro de Hojas de Control	Pantalla	prueba_evaluacion_impresora_inspeccion.jsp
508	Registro de Hojas de Control	Pantalla	prueba_evaluacion_impresora_nivel_homogeneidad.jsp
509	Registro de Hojas de Control	Pantalla	prueba_evaluacion_impresora_referencia.jsp
510	Registro de Hojas de Control	Pantalla	prueba_evaluacion_impresora_referencia_sensitometria.jsp
511	Registro de Hojas de Control	Pantalla	prueba_evaluacion_impresora_sensitometria.jsp
512	Registro de Hojas de Control	Pantalla	prueba_evaluacion_monitores_inspeccion.jsp
513	Registro de Hojas de Control	Pantalla	prueba_evaluacion_monitores_nivel_homogeneidad.jsp
514	Registro de Hojas de Control	Pantalla	prueba_fuerza_compresion.jsp
515	Registro de Hojas de Control	Pantalla	prueba_homogeneidad_receptor_imagen.jsp
516	Registro de Hojas de Control	Pantalla	prueba_pelicula_pantalla.jsp
517	Registro de Hojas de Control	Pantalla	prueba_procesador_pelicula_referencia.jsp
518	Registro de Hojas de Control	Pantalla	prueba_registro_calidad_imagen_actual.jsp
519	Registro de Hojas de Control	Pantalla	prueba_registro_calidad_imagen_referencia.jsp
520	Registro de Hojas de Control	Pantalla	prueba_registro_calidad_imagen_referencia_consulta.jsp
521	Registro de Hojas de Control	Pantalla	prueba_revision_visual_equipo_instalaciones.jsp
522	Registro de Hojas de Control	Pantalla	prueba_sensitometria.jsp
523	Registro de Hojas de Control	Pantalla	titulos_ingresa_instrumento.jsp
524	Registro de Hojas de Control	Pantalla	titulos_prueba.jsp
525	Mantenimiento de Usuario	Pantalla	main_users.jsp
526	Mantenimiento de Usuario	Pantalla	aso_usuario_perfiles.jsp
527	Mantenimiento de Usuario	Pantalla	bus_usuario.jsp
528	Mantenimiento de Usuario	Pantalla	cabecera_usuario.jsp
529	Mantenimiento de Usuario	Pantalla	ctg_usuarios.jsp
530	Mantenimiento de Usuario	Pantalla	ficha_usuario.jsp
531	Seguridad	Pantalla	perfiles.jsp
532	Mantenimiento de Usuario	Pantalla	ficha_usuario.jsp
533	Mantenimiento de Usuario	Pantalla	ing_usuario.jsp
534	Mantenimiento de Usuario	Pantalla	ing_usuario.jsp
535	Mantenimiento de Usuario	Pantalla	codigos_postales.jsp
536	Mantenimiento de Usuario	Pantalla	codigo_postal.jsp
537	Utilería	Pantalla	grafica.jsp
538	Utilería	Pantalla	muestra_imagen.jsp
539	Plantilla del Sistema	Proceso	main_style.css
540	Plantilla del Sistema	Proceso	menu_superior.css
541	Plantilla del Sistema	Proceso	tables.css
542	Plantilla del Sistema	Proceso	tables2.css
543	Plantilla del Sistema	Proceso	visualize-dark.css
544	Plantilla del Sistema	Proceso	visualize-light.css
545	Archivo de configuración	Proceso	visualize.css

546	Archivo de configuración	Proceso	decorator.xml
547	Plantilla del Sistema	Proceso	layout.jsp
548	Plantilla del Sistema	Proceso	layout_main.jsp
549	Archivo de configuración	Proceso	sitemesh.xml
550	Archivo de configuración	Proceso	web.xml

Listatodo de Funciones[6]

2.4 MAPA DE NAVEGACIÓN

En el mapa se diseña la navegación del sistema mostrando a que tendrán acceso los usuarios de acuerdo a su perfil (Administrador, Físico Médico y Técnica).

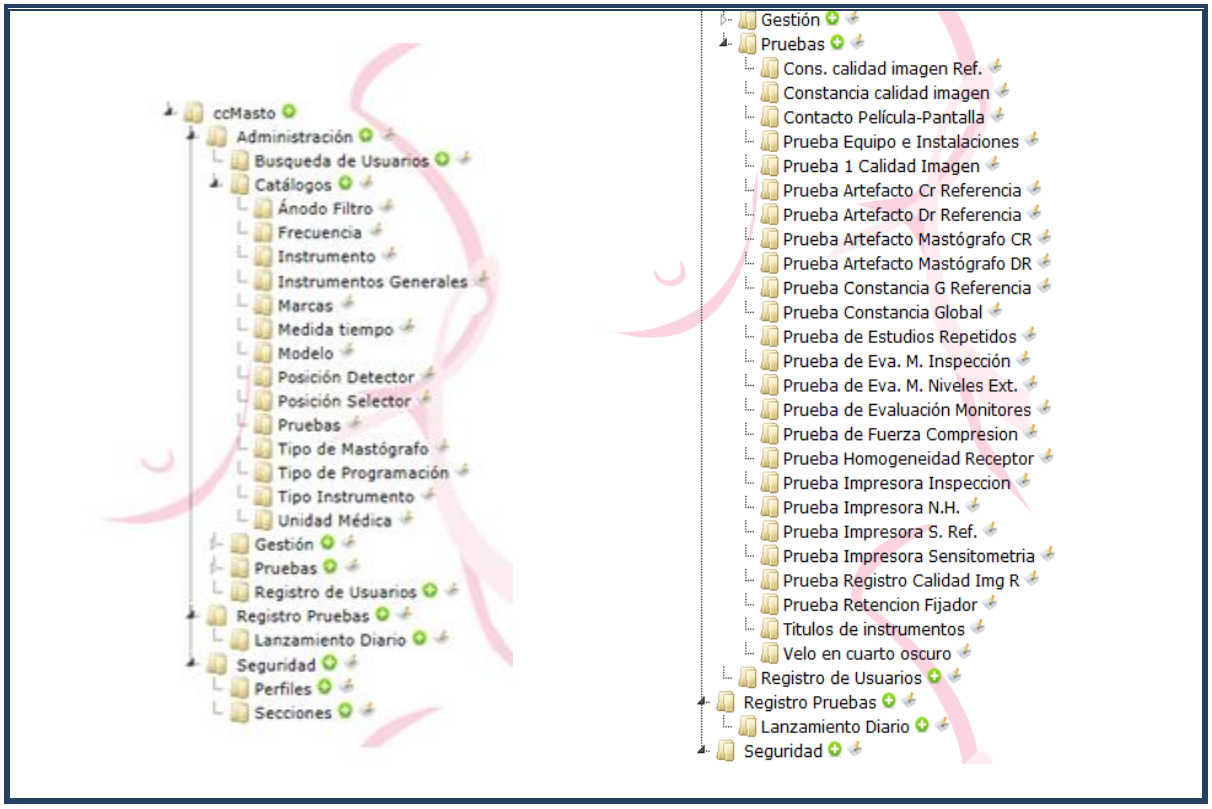


Figura 4.4 ***Elementos del Mapa de Navegación.***

SECCIONES	Administrador	Físico	Técnica
Búsqueda de Usuarios - usuarios	x	x	
Catálogos - Mantenimiento de catálogos	x	x	
Ánodo Filtro - Mantenimiento del catalogo ánodo filtro	x	x	
Frecuencia - Catálogo de frecuencias	x	x	

[6] Listado Completo de la funciones que se realizaron para la construcción de la solución propuesta.

Instrumento - Catálogo de instrumentos	x	x	
Instrumentos Generales - Registro de instrumentos asociados a la unidad	x	x	
Marcas - Mantenimiento del catalogo de marcas	x	x	
Medida tiempo - Catalogo de medidas de tiempo	x	x	
Modelo - Mantenimiento del catalogo de modelos	x	x	
Posición Detector - Registro de posición detector	x	x	
Posición Detector - Descripción de Posición Detector	x	x	
Posición Selector - Registro de posición selector	x	x	
Pruebas - Catalogo de pruebas	x	x	
Tipo de Mastógrafos - Mantenimiento del catalogo de tipo de mastógrafos	x	x	
Tipo de Programación - Mantenimiento de catalogo de tipo de programación	x	x	
Tipo Instrumento - Catalogo de tipo de instrumentos	x	x	
Unidad Médica - Catálogo de unidades médicas	x	x	
Gestión - Menú de gestión de pruebas	x	x	
Mapa - Mapa	x	x	x
Monitoreo (Calendario) - Calendario para monitoreo de pruebas	x	x	
Parámetros de Prueba - Configuración de los parámetros de las pruebas	x	x	
Programación - Programación de pruebas	x	x	
Pruebas x Unidades - Asociación de pruebas por unidades	x	x	
Pruebas - Pantalla de lanzamiento de pruebas	x	x	x
Cons. calidad imagen Ref. - Cons. calidad imagen Ref.	x	x	x
Constancia calidad imagen - Constancia de la calidad de la imagen	x	x	x
Contacto Película-Pantalla - Hoja de registro de contacto película-pantalla	x	x	x
Prueba Equipo e Instalaciones - Registro Equipamiento e Instalaciones	x	x	x
Prueba 1 Calidad Imagen - Registro 1 Calidad Imagen	x	x	x
Prueba Artefacto Cr Referencia - Registro de artefactos en mastógrafos Cr NR	x	x	x
Prueba Artefacto Dr Referencia - Registro de artefactos en mastógrafos DR NR	x	x	x
Prueba Artefacto Mastógrafos CR - Registro de artefactos en mastógrafos CR	x	x	x
Prueba Artefacto Mastógrafos DR - Registro de artefactos en mastógrafos DR	x	x	x
Prueba Constancia G Referencia - Registro de Constancia Global Referencia	x	x	x
Prueba Constancia Global - Registro de Constancia Global	x	x	x
Prueba de Estudios Repetidos - Registro de pruebas de estudios repetidos	x	x	x
Prueba de Eva. M. Inspección - Registro de Eva. M. Inspección	x	x	x
Prueba de Eva. M. Niveles Ext. - Registro de Eva. M. Niveles Ext.	x	x	x
Prueba de Evaluación Monitores - Registro evaluación monitores	x	x	x
Prueba de Fuerza Compresión - Registro de la prueba de Fuerza de Compresión	x	x	x
Prueba Homogeneidad Receptor - Registros de Homogeneidad del Receptor	x	x	x
Prueba Impresora Inspección - Registro Impresora Inspección	x	x	x
Prueba Impresora N.H. - Registro de Impresora N.H.	x	x	x
Prueba Impresora S. Ref. - Registro Impresora S. Ref.	x	x	x

Prueba Impresora Sensitometría - Registro Impresora Sensitometría	x	x	x
Prueba Registro Calidad Img. R - Registro Calidad Img. Ref.	x	x	x
Prueba Retención Fijador - Registro de Retención Fijador	x	x	x
Títulos de instrumentos - Cabecera general de Instrumentos	x	x	x
Velo en cuarto oscuro - Registro de velo en cuarto oscuro	x	x	x
Registro de Usuarios - Registro de un nuevo de usuario	x	x	x
Registro Pruebas - Menú de registro de pruebas	x	x	x
Lanzamiento Diario - Lanzamiento diario de pruebas	x	x	x
Seguridad - Administra la seguridad del sistema	x		
Perfiles - perfiles	x		
Secciones - secciones	x		

3. DESARROLLO

El desarrollo pertenece a la fase de Construcción en la metodología implementada, y consiste de las siguientes tareas:

1. Clarificación de los requerimientos,
2. Codificación de los programas,
3. Pruebas con datos muestra y
4. Solución de problemas detectados.

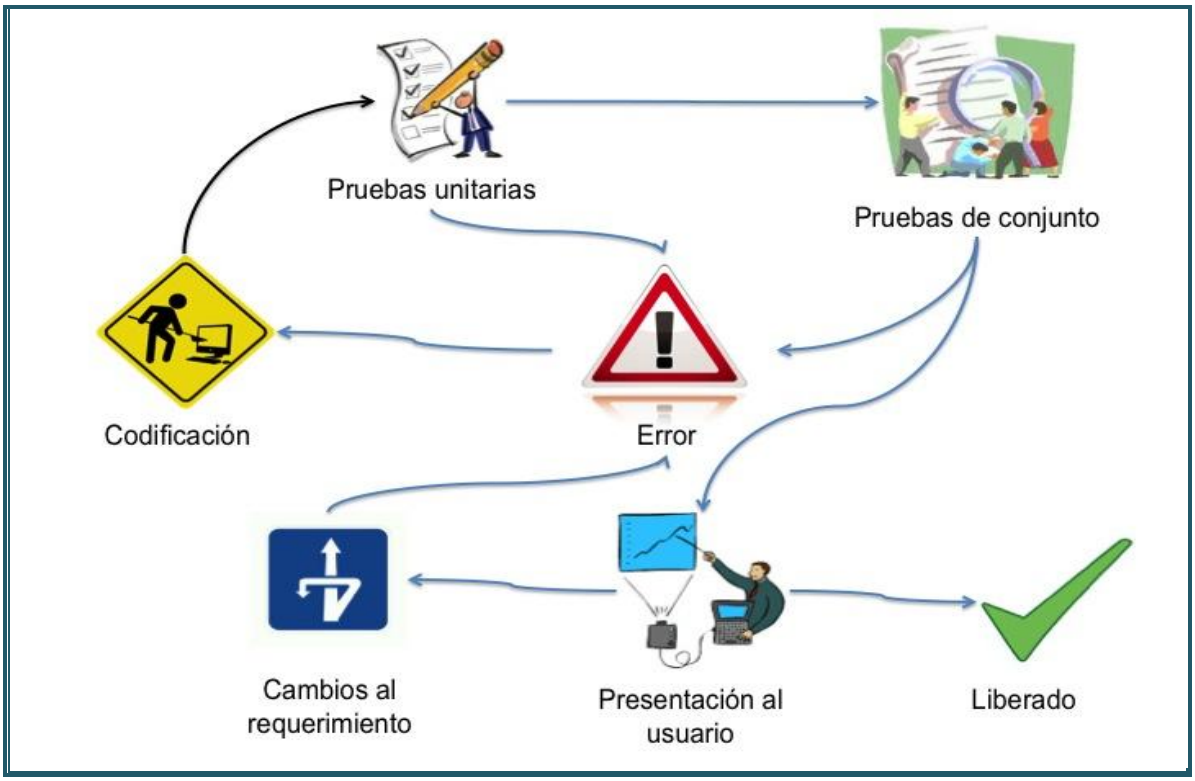


Figura 4.5 ***Diagrama de actividades del desarrollo de la solución propuesta.***

A continuación se explica cómo se realizó cada una de estas tareas para el desarrollo del sistema.

3.1 CLARIFICACIÓN DE REQUERIMIENTOS CONSTRUIDOS

Una parte importante en el desarrollo del sistema consiste en que el usuario y el desarrollador estén de acuerdo en lo que se va a crear. Desde el punto de vista del usuario, debe quedar claro que lo que el desarrollador va a hacer es lo que el usuario está pidiendo. Muchas veces a los usuarios se les dificulta el poder visualizar como va a quedar el sistema y aquí los desarrolladores hacemos uso de la construcción de prototipos.

En nuestro caso los prototipos solo fueron dibujados en powerpoint para mostrar a los usuarios como podrían quedar algunas de las pantallas. En una junta se mostraron a los usuarios las pantallas y todas sus dudas de operación fueron contestadas en esa reunión. De igual forma, en esa junta se aclararon varias dudas de funcionamiento que sirvieron para verificar los puntos más complejos del sistema.

Al terminar la junta se entregó el documento con el prototipo a los usuarios y éstos firmaron para indicar que estaban de acuerdo con el mismo.

Una vez que se comenzó con la construcción del sistema, cuando ya habíamos terminado el desarrollo de algunos catálogos, se agendó una visita para que los usuarios conocieran finalmente la forma en que las pantallas funcionarían. Con esto se pretendía que los usuarios entendieran como iban a trabajar dichas pantallas y que estuvieran de acuerdo, así como que se hicieran observaciones y comentarios para integrarlos a las pantallas posteriores.

Uno de los primeros comentarios fue el de modificar el fondo de pantalla y poner uno más acorde al **CNEGySR**. También surgieron muchas dudas y preguntas acerca del funcionamiento de las pantallas y los catálogos, las cuales se explicaron en su momento. De ahí en adelante se calendarizaron juntas semanales para que se pudieran revisar los avances del proyecto.

En reuniones posteriores todo fue avanzando conforme a lo esperado y no hubo muchas modificaciones del sistema, hasta que se llego a la parte del desarrollo de las pruebas, que es cuando se empezaron a presentar los problemas. Uno de los principales problemas es que el desarrollo del sistema se hizo en paralelo a las actualizaciones del manual de control de calidad por parte de los usuarios finales, dando como resultado que los protocolos cambiaran la mayor parte de las pruebas a desarrollar. Los cambios anteriores se nos eran avisados hasta que nos presentábamos en su oficina para mostrar el avance del sistema. Esta forma de comunicación tuvo como resultado que la mayoría de las pruebas fueran cambiadas una vez que el desarrollo estaba finalizado, lo cual implicó atrasos en el mismo.

En diversas ocasiones debido a que las pruebas ya habían cambiado, a nosotros nos surgían dudas de cómo debería funcionar un requerimiento y lo que hacíamos era comunicarnos con los usuarios para que nos aclararan el funcionamiento de acuerdo al nuevo manual de control de calidad, el cual nos era proporcionado por los mismos usuarios vía correo electrónico. También, cuando nuestras dudas coincidían con una junta de presentación del sistema, entonces estas eran aclaradas de forma personal, y en muchos casos, también permitían que los usuarios finales se percataran de que su procedimiento era erróneo y realizaran las modificaciones correspondientes.

En conclusión, se puede decir que la clarificación de requerimientos nos sirvió a ambas partes para afinar detalles y así dejar un sistema más robusto, sin embargo el costo fue que los tiempos de desarrollo se incrementaran debido a la falta de comunicación que los usuarios tenían con nosotros.

3.2 CODIFICACIÓN DE LOS PROGRAMAS

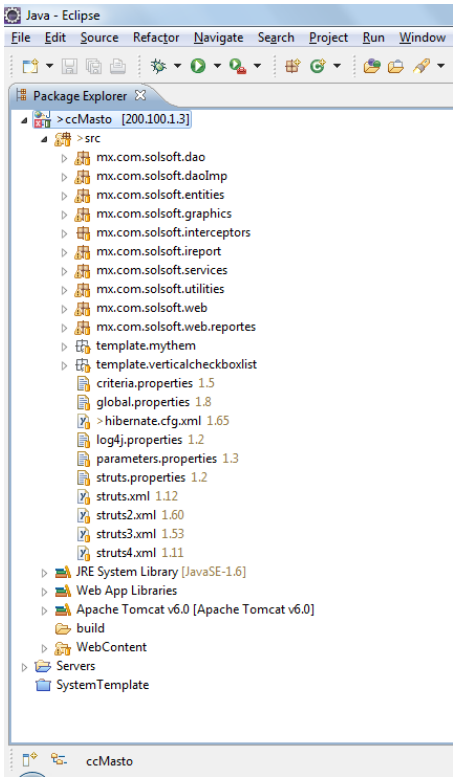
El desarrollo del sistema se empezó a hacer de acuerdo al plan de trabajo que se elaboró en la etapa de análisis, y se trató de hacer que la repartición fuera equitativa según la cantidad de días que se habían estimado originalmente, sin embargo, en algunos programas nos dimos cuenta que la estimación había estado mal calculada, o en algunos casos el requerimiento inicial cambió, lo que hizo que él tiempo de desarrollo se incrementara. La repartición del sistema quedó de la siguiente forma:

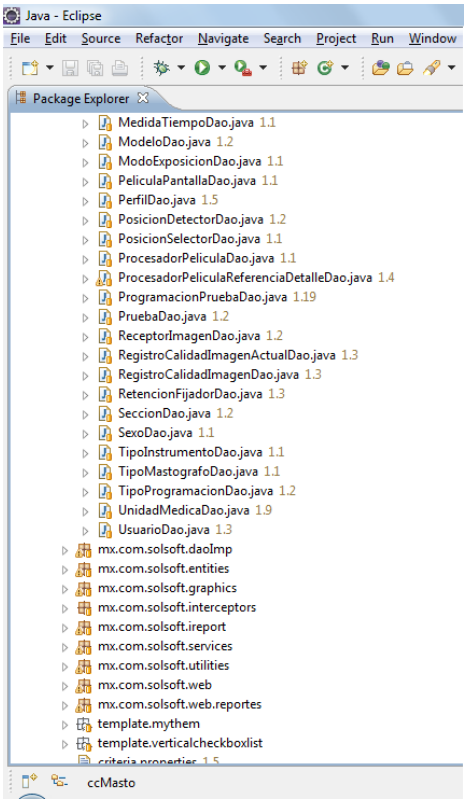
Módulo	Requerimientos	Consultor
Catálogos		
	Pantalla de mantenimiento de entidad federativa	AFP
	Pantalla de mantenimiento de Unidad Médica	AFP
	Pantalla de mantenimiento de Prueba	AFP
	Pantalla de mantenimiento de medida de tiempo	AFP
	Pantalla de mantenimiento de frecuencia	AFP
	Pantalla de mantenimiento de tipo de instrumento	AFP
	Pantalla de mantenimiento de Instrumento	AFP
	Pantalla de mantenimiento Posición control de densidad CAE	AFP
	Pantalla de mantenimiento Modo CAE	AFP
	Pantalla de mantenimiento Modo de exposición	AFP
	Pantalla de mantenimiento de tipo de mastografo	EMM
	Pantalla de mantenimiento de ánodo/filtro	EMM
	Asociación de tipos mastógrafo con ánodo/filtro	EMM
	Pantalla de mantenimiento de marca	EMM
	Pantalla de mantenimiento de modelo	EMM
	Asociación de tipos de instrumento con marcas	EMM
	Pantalla de mantenimiento de tipo de programación	EMM
	Pantalla de mantenimiento Posicion CAE	EMM
	Pantalla de mantenimiento Posición del detector del CAE	EMM
	Pantalla de mantenimiento Posición del selector de densidad del CAE	EMM
Gestión de pruebas		
	Pantalla de definición de pruebas	AFP
	Pantalla de asociaciación de definición de pruebas con unidades médicas	AFP
	Pantalla de asociación de instrumentos con definición de prueba	AFP
	Proceso de programación de la prueba	AFP
	Reporte de Resumen de control	AFP
	Actualizar estado de las pruebas	AFP
	Monitorear pruebas	AFP
	Pantalla de generación de pruebas	AFP
	Proceso de generación de pruebas	AFP
	Pantalla de calendario de monitoreo de pruebas	AFP
Capturar pruebas		
	Cabecera de pruebas	EMM
	Pantalla de Evaluación de Fuerza de compresión	EMM
	Pantalla de Análisis de Estudios Repetidos	EMM
	Pantalla de Revisión visual del Equipamiento e Instalaciones	EMM
	Pantalla de Constancia en la Homogeneidad del Receptor	EMM
	Pantalla de Constancia de Funcionamiento Global del CAE	EMM
	Pantalla de Artefactos en mastógrafos DR y CR	EMM
	Pantalla de Evaluación de Monitores	AFP
	Pantalla de Evaluación de Impresoras	AFP
	Pantalla de Constancia de la Calidad de la Imagen	AFP
	Pantalla de Procesador de Película	EMM
	Pantalla de Calidad de Imagen	EMM
	Pantalla de Cuarto Oscuro	EMM
	Pantalla de Contacto película-pantalla	AFP
	Pantalla de Análisis de Retención del Fijador en la Película	AFP
	Impresión de Evaluación de Fuerza de compresión	AFP
	Impresión de Análisis de Estudios Repetidos	EMM
	Impresión de Revisión visual del Equipamiento e Instalaciones	EMM
	Impresión de Análisis de Retención del Fijador en la Película	EMM
	Impresión de Constancia de Funcionamiento Global del CAE	EMM
	Impresión de Constancia en la Homogeneidad del Receptor	AFP
	Impresión de Artefactos en mastógrafos DR y CR	AFP
	Impresión de Constancia de la Calidad de la Imagen	AFP
	Impresión de Evaluación de Monitores	AFP
	Impresión de Evaluación de Impresoras	EMM
	Graficar pruebas de Procesador de Película	EMM
	Graficar pruebas de Calidad de Imagen	EMM
	Graficar pruebas de Constancia de Funcionamiento Global del CAE	AFP
	Graficar pruebas de Constancia de la Calidad de la Imagen	AFP
	Graficar pruebas de Evaluación de Impresoras	AFP
Control de Acceso		
	Pantalla de control de acceso	EMM
	Proceso de verificación de usuario	EMM
Administración de usuarios		
	Pantalla de mantenimiento de usuarios	EMM
	Asociar perfil al usuario	EMM
	Asociar unidad al usuario	EMM
	Pantalla de mantenimiento de perfiles	EMM

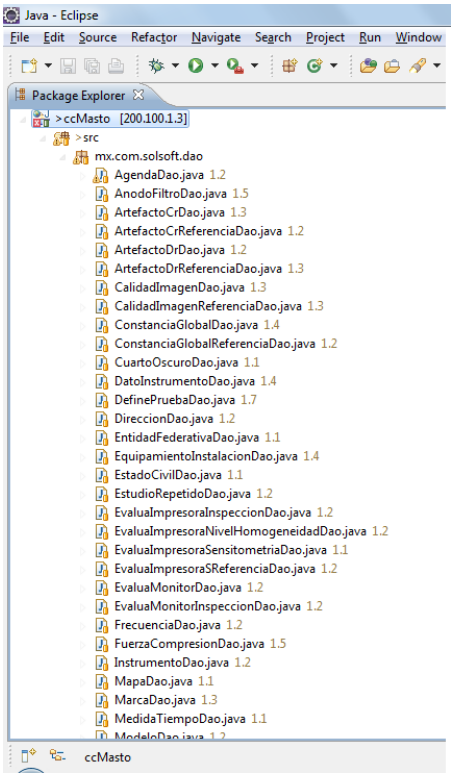
AFP – Agustín Fuentevilla Perera, EMM – Elizabeth Morales Morato[7]

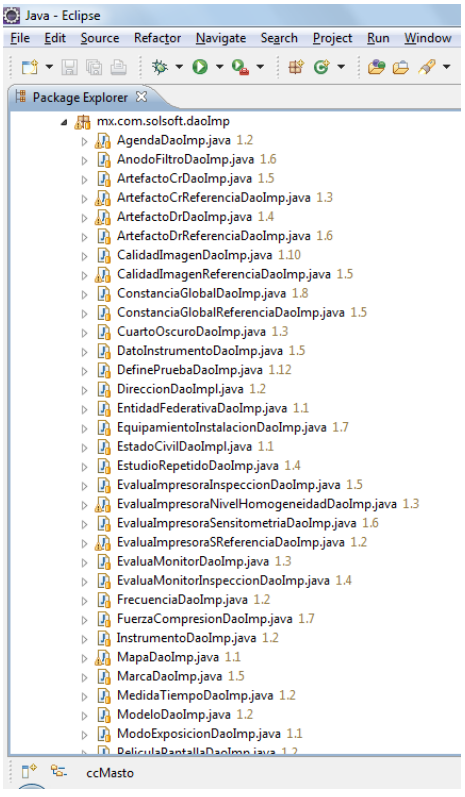
[7] Listado de asignación de los requerimientos propuestos.

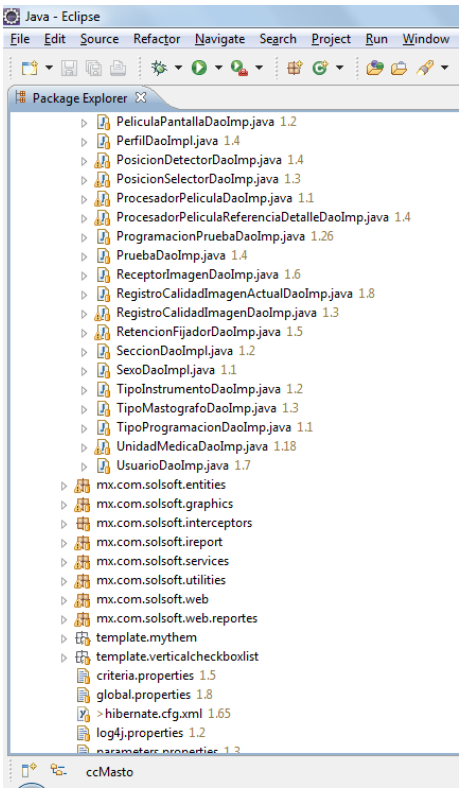
A continuación se muestran los programas desarrollados para el sistema:

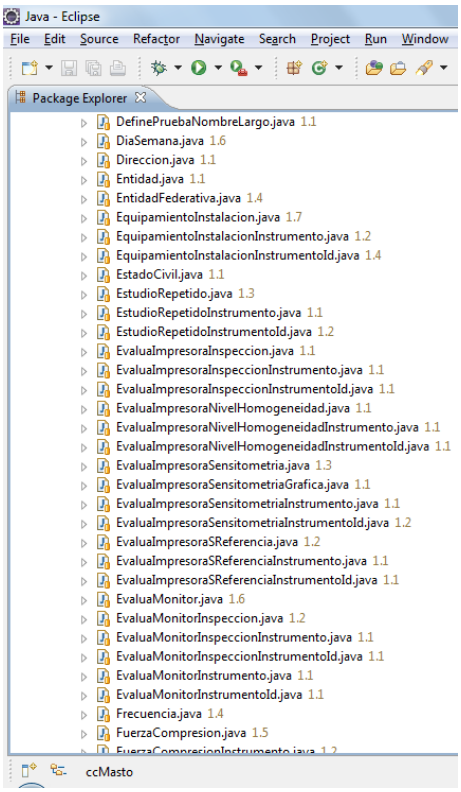


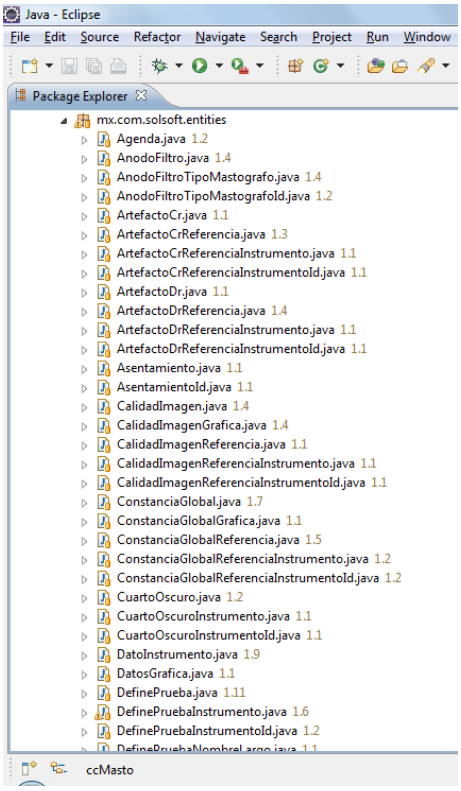


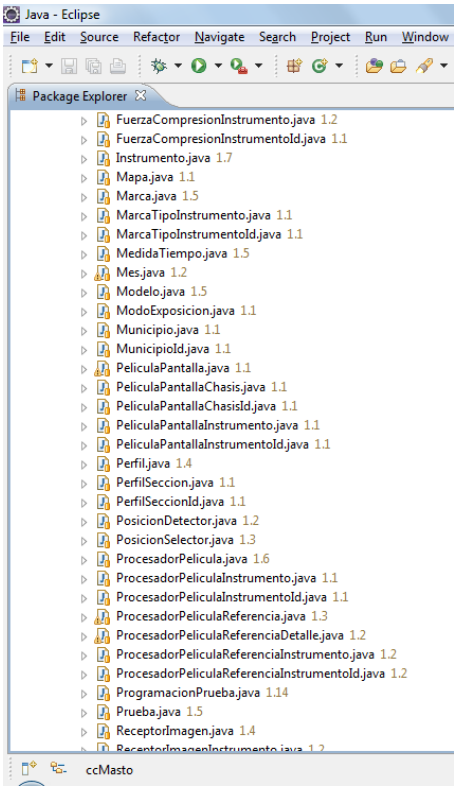


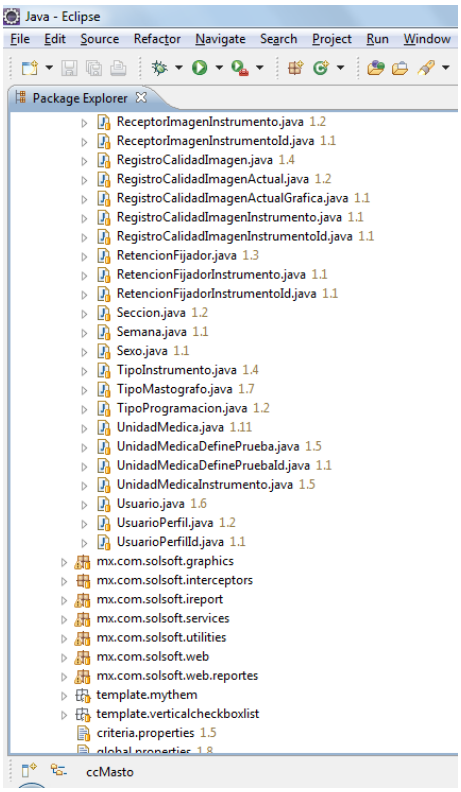


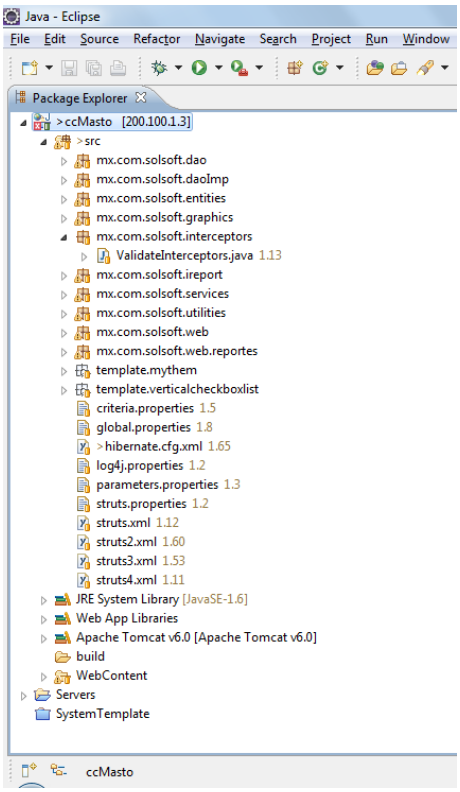

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
PeliculaPantallaDaoImp.java 1.2
PerfilDaoImpl.java 1.4
PosicionDetectorDaoImp.java 1.4
PosicionSelectorDaoImp.java 1.3
ProcesadorPeliculaDaoImp.java 1.1
ProcesadorPeliculaReferenciaDetalleDaoImp.java 1.4
ProgramacionPruebaDaoImp.java 1.26
PruebaDaoImp.java 1.4
ReceptorImagenDaoImp.java 1.6
RegistroCalidadImagenActualDaoImp.java 1.8
RegistroCalidadImagenDaoImp.java 1.3
RetencionFijadorDaoImp.java 1.5
SeccionDaoImpl.java 1.2
SexoDaoImpl.java 1.1
TipoInstrumentoDaoImp.java 1.2
TipoMastografoDaoImp.java 1.3
TipoProgramacionDaoImp.java 1.1
UnidadMedicaDaoImp.java 1.18
UsuarioDaoImp.java 1.7
mx.com.solsoft.entities
mx.com.solsoft.graphics
mx.com.solsoft.interceptors
mx.com.solsoft.ireport
mx.com.solsoft.services
mx.com.solsoft.utilities
mx.com.solsoft.web
mx.com.solsoft.web.reportes
template.mythem
template.verticalcheckboxlist
criteria.properties 1.5
global.properties 1.8
hibernate.cfg.xml 1.65
log4j.properties 1.2
ccMasto

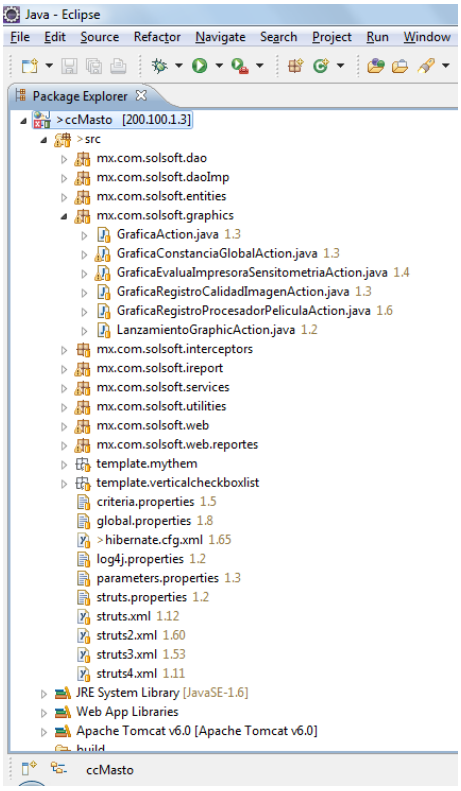

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
DefinePruebaNombreLargo.java 1.1
DiaSemana.java 1.6
Direccion.java 1.1
Entidad.java 1.1
EntidadFederativa.java 1.4
EquipamientoInstalacion.java 1.7
EquipamientoInstalacionInstrumento.java 1.2
EquipamientoInstalacionInstrumentoId.java 1.4
EstadoCivil.java 1.1
EstudioRepetido.java 1.3
EstudioRepetidoInstrumento.java 1.1
EstudioRepetidoInstrumentoId.java 1.2
EvaluaImpresoraInspeccion.java 1.1
EvaluaImpresoraInspeccionInstrumento.java 1.1
EvaluaImpresoraInspeccionInstrumentoId.java 1.1
EvaluaImpresoraNivelHomogeneidad.java 1.1
EvaluaImpresoraNivelHomogeneidadInstrumento.java 1.1
EvaluaImpresoraNivelHomogeneidadInstrumentoId.java 1.1
EvaluaImpresoraSensitometria.java 1.3
EvaluaImpresoraSensitometriaGrafica.java 1.1
EvaluaImpresoraSensitometriaInstrumento.java 1.1
EvaluaImpresoraSensitometriaInstrumentoId.java 1.2
EvaluaImpresoraSReferencia.java 1.2
EvaluaImpresoraSReferenciaInstrumento.java 1.1
EvaluaImpresoraSReferenciaInstrumentoId.java 1.1
EvaluaMonitor.java 1.6
EvaluaMonitorInspeccion.java 1.2
EvaluaMonitorInspeccionInstrumento.java 1.1
EvaluaMonitorInspeccionInstrumentoId.java 1.1
EvaluaMonitorInstrumento.java 1.1
EvaluaMonitorInstrumentoId.java 1.1
Frecuencia.java 1.4
FuerzaCompresion.java 1.5
ccMasto

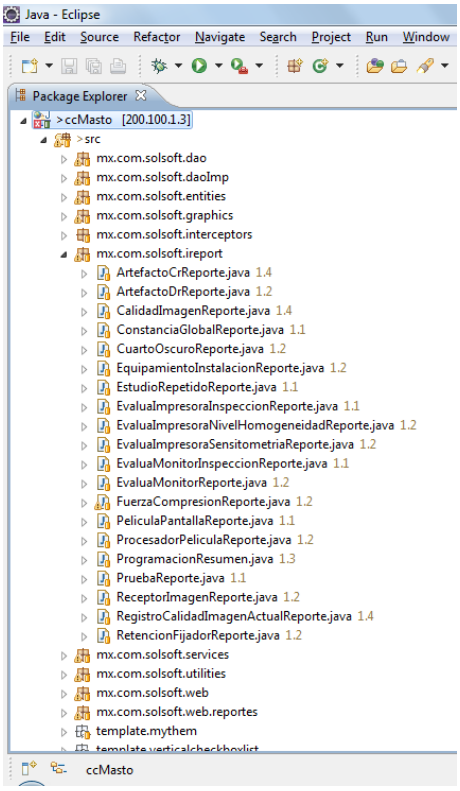

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
mx.com.solsoft.entities
Agenda.java 1.2
AnodoFiltro.java 1.4
AnodoFiltroTipoMastografo.java 1.4
AnodoFiltroTipoMastografoId.java 1.2
ArtefactoCr.java 1.1
ArtefactoCrReferencia.java 1.3
ArtefactoCrReferenciaInstrumento.java 1.1
ArtefactoCrReferenciaInstrumentoId.java 1.1
ArtefactoDr.java 1.1
ArtefactoDrReferencia.java 1.4
ArtefactoDrReferenciaInstrumento.java 1.1
ArtefactoDrReferenciaInstrumentoId.java 1.1
Asentamiento.java 1.1
AsentamientoId.java 1.1
CalidadImagen.java 1.4
CalidadImagenGrafica.java 1.4
CalidadImagenReferencia.java 1.1
CalidadImagenReferenciaInstrumento.java 1.1
CalidadImagenReferenciaInstrumentoId.java 1.1
ConstanciaGlobal.java 1.7
ConstanciaGlobalGrafica.java 1.1
ConstanciaGlobalReferencia.java 1.5
ConstanciaGlobalReferenciaInstrumento.java 1.2
ConstanciaGlobalReferenciaInstrumentoId.java 1.2
CuartoOscuro.java 1.2
CuartoOscuroInstrumento.java 1.1
CuartoOscuroInstrumentoId.java 1.1
DatoInstrumento.java 1.9
DatosGrafica.java 1.1
DefinePrueba.java 1.11
DefinePruebaInstrumento.java 1.6
DefinePruebaInstrumentoId.java 1.2
ccMasto

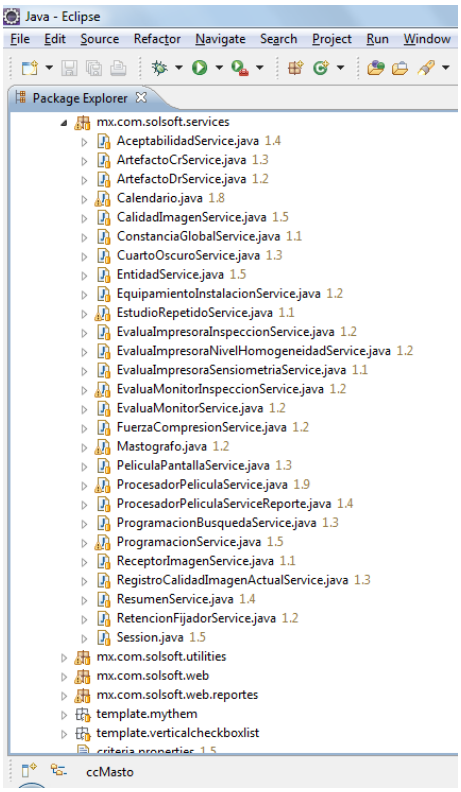

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
FuerzaCompresionInstrumento.java 1.2
FuerzaCompresionInstrumentoId.java 1.1
Instrumento.java 1.7
Mapa.java 1.1
Marca.java 1.5
MarcaTipoInstrumento.java 1.1
MarcaTipoInstrumentoId.java 1.1
MedidaTiempo.java 1.5
Mes.java 1.2
Modelo.java 1.5
ModoExposicion.java 1.1
Municipio.java 1.1
MunicipioId.java 1.1
PeliculaPantalla.java 1.1
PeliculaPantallaChasis.java 1.1
PeliculaPantallaChasisId.java 1.1
PeliculaPantallaInstrumento.java 1.1
PeliculaPantallaInstrumentoId.java 1.1
Perfil.java 1.4
PerfilSeccion.java 1.1
PerfilSeccionId.java 1.1
PosicionDetector.java 1.2
PosicionSelector.java 1.3
ProcesadorPelicula.java 1.6
ProcesadorPeliculaInstrumento.java 1.1
ProcesadorPeliculaInstrumentoId.java 1.1
ProcesadorPeliculaReferencia.java 1.3
ProcesadorPeliculaReferenciaDetalle.java 1.2
ProcesadorPeliculaReferenciaInstrumento.java 1.2
ProcesadorPeliculaReferenciaInstrumentoId.java 1.2
ProgramacionPrueba.java 1.14
Prueba.java 1.5
ReceptorImagen.java 1.4
ccMasto

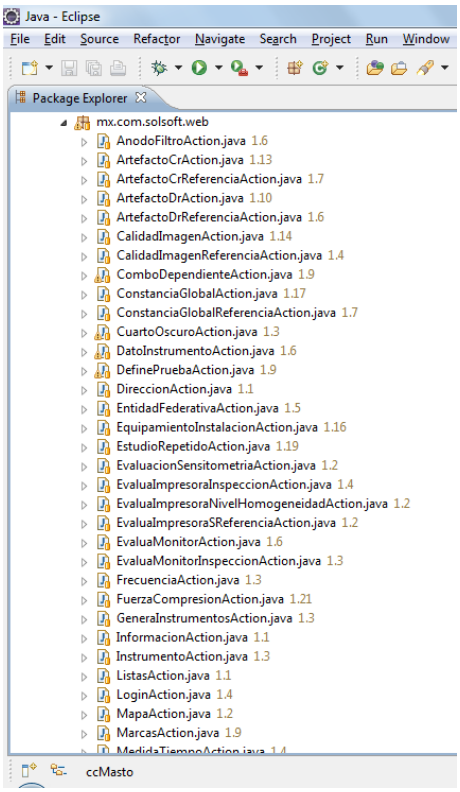

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
ReceptorImagenInstrumento.java 1.2
ReceptorImagenInstrumentoId.java 1.1
RegistroCalidadImagen.java 1.4
RegistroCalidadImagenActual.java 1.2
RegistroCalidadImagenActualGrafica.java 1.1
RegistroCalidadImagenInstrumento.java 1.1
RegistroCalidadImagenInstrumentoId.java 1.1
RetencionFijador.java 1.3
RetencionFijadorInstrumento.java 1.1
RetencionFijadorInstrumentoId.java 1.1
Seccion.java 1.2
Semana.java 1.1
Sexo.java 1.1
TipoInstrumento.java 1.4
TipoMastografo.java 1.7
TipoProgramacion.java 1.2
UnidadMedica.java 1.11
UnidadMedicaDefinePrueba.java 1.5
UnidadMedicaDefinePruebaId.java 1.1
UnidadMedicaInstrumento.java 1.5
Usuario.java 1.6
UsuarioPerfil.java 1.2
UsuarioPerfilId.java 1.1
mx.com.solsoft.graphics
mx.com.solsoft.interceptors
mx.com.solsoft.ireport
mx.com.solsoft.services
mx.com.solsoft.utilities
mx.com.solsoft.web
mx.com.solsoft.web.reportes
template.mythem
template.verticalcheckboxlist
criteria.properties 1.5
ccMasto

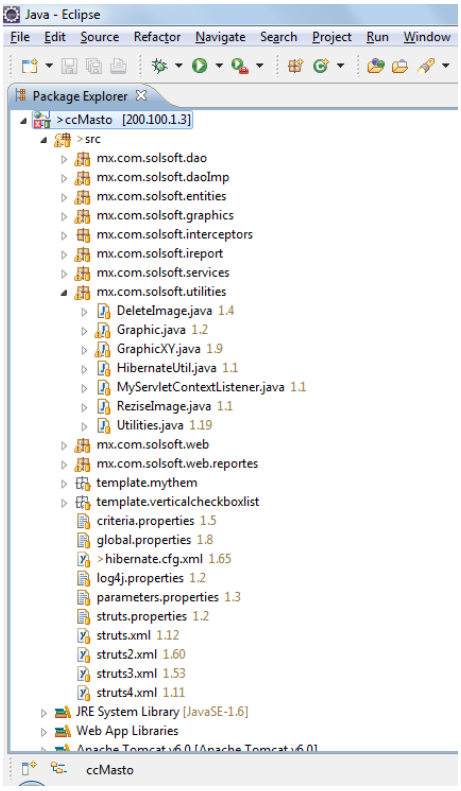

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
>ccMasto [200.100.1.3]
>src
mx.com.solsoft.dao
mx.com.solsoft.daoImp
mx.com.solsoft.entities
mx.com.solsoft.graphics
mx.com.solsoft.interceptors
ValidateInterceptors.java 1.13
mx.com.solsoft.ireport
mx.com.solsoft.services
mx.com.solsoft.utilities
mx.com.solsoft.web
mx.com.solsoft.web.reportes
template.mythem
template.verticalcheckboxlist
criteria.properties 1.5
global.properties 1.8
>hibernate.cfg.xml 1.65
log4j.properties 1.2
parameters.properties 1.3
struts.properties 1.2
struts.xml 1.12
struts2.xml 1.60
struts3.xml 1.53
struts4.xml 1.11
JRE System Library [JavaSE-1.6]
Web App Libraries
Apache Tomcat v6.0 [Apache Tomcat v6.0]
build
WebContent
Servers
SystemTemplate
ccMasto

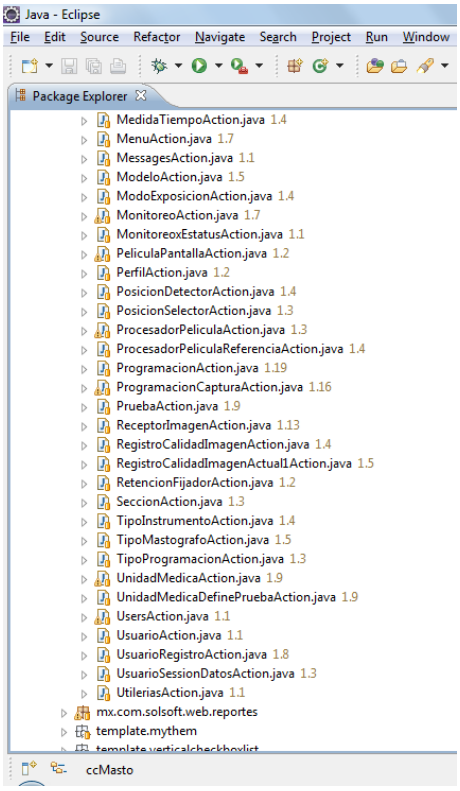

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
>ccMasto [200.100.1.3]
>src
mx.com.solsoft.dao
mx.com.solsoft.daoImp
mx.com.solsoft.entities
mx.com.solsoft.graphics
GraficaAction.java 1.3
GraficaConstanciaGlobalAction.java 1.3
GraficaEvaluaImpresoraSensitometriaAction.java 1.4
GraficaRegistroCalidadImagenAction.java 1.3
GraficaRegistroProcesadorPeliculaAction.java 1.6
LanzamientoGraphicAction.java 1.2
mx.com.solsoft.interceptors
mx.com.solsoft.ireport
mx.com.solsoft.services
mx.com.solsoft.utilities
mx.com.solsoft.web
mx.com.solsoft.web.reportes
template.mythem
template.verticalcheckboxlist
criteria.properties 1.5
global.properties 1.8
>hibernate.cfg.xml 1.65
log4j.properties 1.2
parameters.properties 1.3
struts.properties 1.2
struts.xml 1.12
struts2.xml 1.60
struts3.xml 1.53
struts4.xml 1.11
JRE System Library [JavaSE-1.6]
Web App Libraries
Apache Tomcat v6.0 [Apache Tomcat v6.0]
ccMasto

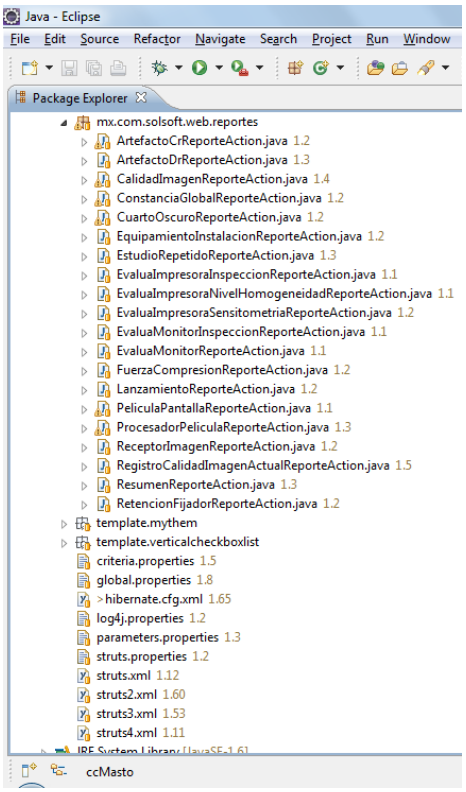

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
>ccMasto [200.100.1.3]
>src
mx.com.solsoft.dao
mx.com.solsoft.daoImp
mx.com.solsoft.entities
mx.com.solsoft.graphics
mx.com.solsoft.interceptors
mx.com.solsoft.ireport
ArtefactoCrReporte.java 1.4
ArtefactoDrReporte.java 1.2
CalidadImagenReporte.java 1.4
ConstanciaGlobalReporte.java 1.1
CuartoOscuroReporte.java 1.2
EquipamientoInstalacionReporte.java 1.2
EstudioRepetidoReporte.java 1.1
EvaluaImpresoraInspeccionReporte.java 1.1
EvaluaImpresoraNivelHomogeneidadReporte.java 1.2
EvaluaImpresoraSensitometriaReporte.java 1.2
EvaluaMonitorInspeccionReporte.java 1.1
EvaluaMonitorReporte.java 1.2
FuerzaCompresionReporte.java 1.2
PeliculaPantallaReporte.java 1.1
ProcesadorPeliculaReporte.java 1.2
ProgramacionResumen.java 1.3
PruebaReporte.java 1.1
ReceptorImagenReporte.java 1.2
RegistroCalidadImagenActualReporte.java 1.4
RetencionFijadorReporte.java 1.2
mx.com.solsoft.services
mx.com.solsoft.utilities
mx.com.solsoft.web
mx.com.solsoft.web.reportes
template.mythem
ccMasto

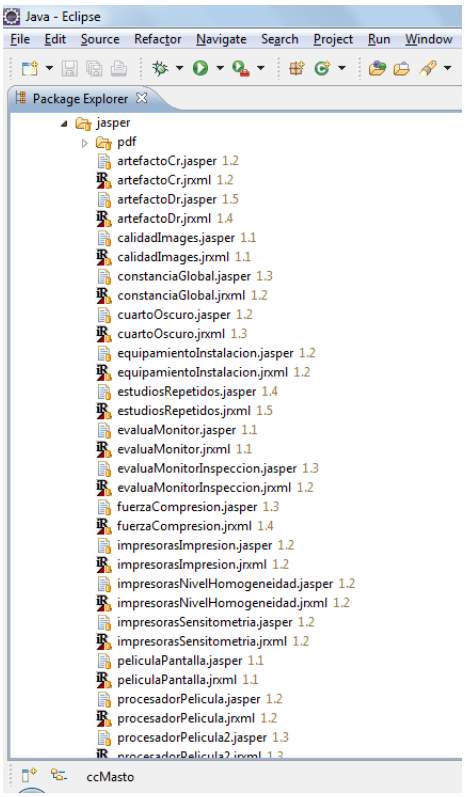

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
mx.com.solsoft.services
AceptabilidadService.java 1.4
ArtefactoCrService.java 1.3
ArtefactoDrService.java 1.2
Calendario.java 1.8
CalidadImagenService.java 1.5
ConstanciaGlobalService.java 1.1
CuartoOscuroService.java 1.3
EntidadService.java 1.5
EquipamientoInstalacionService.java 1.2
EstudioRepetidoService.java 1.1
EvaluaImpresoraInspeccionService.java 1.2
EvaluaImpresoraNivelHomogeneidadService.java 1.2
EvaluaImpresoraSensiometriaService.java 1.1
EvaluaMonitorInspeccionService.java 1.2
EvaluaMonitorService.java 1.2
FuerzaCompresionService.java 1.2
Mastografo.java 1.2
PeliculaPantallaService.java 1.3
ProcesadorPeliculaService.java 1.9
ProcesadorPeliculaServiceReporte.java 1.4
ProgramacionBusquedaService.java 1.3
ProgramacionService.java 1.5
ReceptorImagenService.java 1.1
RegistroCalidadImagenActualService.java 1.3
ResumenService.java 1.4
RetencionFijadorService.java 1.2
Session.java 1.5
mx.com.solsoft.utilities
mx.com.solsoft.web
mx.com.solsoft.web.reportes
template.mythem
template.verticalcheckboxlist
ccMasto

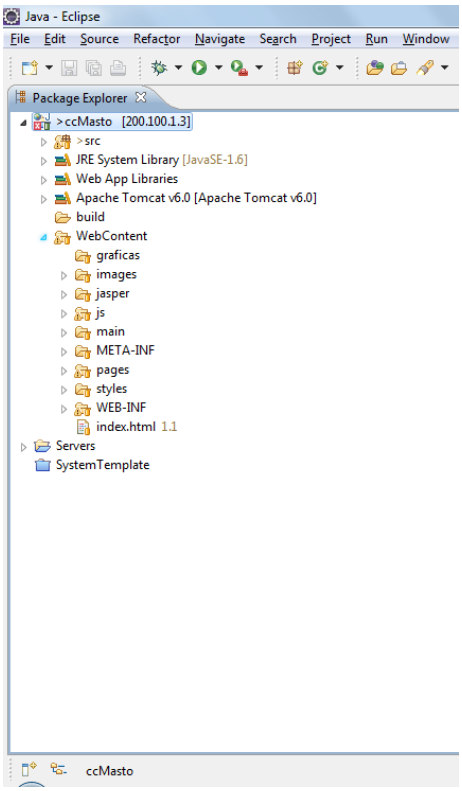

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
mx.com.solsoft.web
AnodoFiltroAction.java 1.6
ArtefactoCrAction.java 1.13
ArtefactoCrReferenciaAction.java 1.7
ArtefactoDrAction.java 1.10
ArtefactoDrReferenciaAction.java 1.6
CalidadImagenAction.java 1.14
CalidadImagenReferenciaAction.java 1.4
ComboDependienteAction.java 1.9
ConstanciaGlobalAction.java 1.17
ConstanciaGlobalReferenciaAction.java 1.7
CuartoOscuroAction.java 1.3
DatoInstrumentoAction.java 1.6
DefinePruebaAction.java 1.9
DireccionAction.java 1.1
EntidadFederativaAction.java 1.5
EquipamientoInstalacionAction.java 1.16
EstudioRepetidoAction.java 1.19
EvaluacionSensitometriaAction.java 1.2
EvaluaImpresoraInspeccionAction.java 1.4
EvaluaImpresoraNivelHomogeneidadAction.java 1.2
EvaluaImpresoraSReferenciaAction.java 1.2
EvaluaMonitorAction.java 1.6
EvaluaMonitorInspeccionAction.java 1.3
FrecuenciaAction.java 1.3
FuerzaCompresionAction.java 1.21
GeneraInstrumentosAction.java 1.3
InformacionAction.java 1.1
InstrumentoAction.java 1.3
ListasAction.java 1.1
LoginAction.java 1.4
MapaAction.java 1.2
MarcasAction.java 1.9
ccMasto

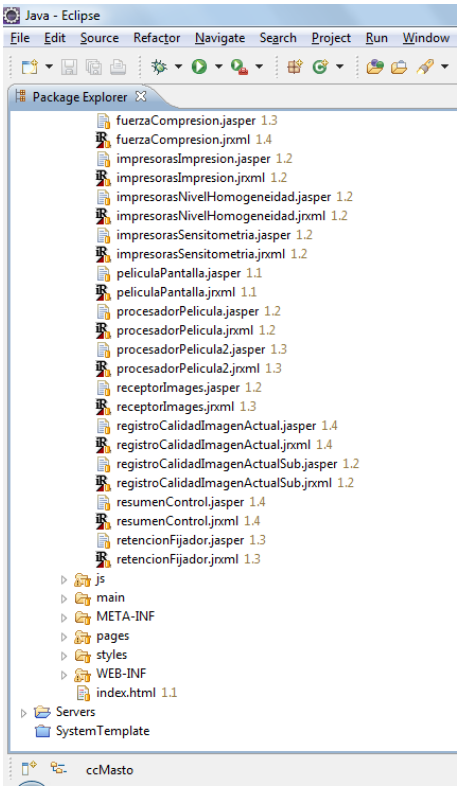

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
>ccMasto [200.100.1.3]
>src
mx.com.solsoft.dao
mx.com.solsoft.daoImp
mx.com.solsoft.entities
mx.com.solsoft.graphics
mx.com.solsoft.interceptors
mx.com.solsoft.ireport
mx.com.solsoft.services
mx.com.solsoft.utilities
DeleteImage.java 1.4
Graphic.java 1.2
GraphicXY.java 1.9
HibernateUtil.java 1.1
MyServletContextListener.java 1.1
ReziseImage.java 1.1
Utilities.java 1.19
mx.com.solsoft.web
mx.com.solsoft.web.reportes
template.mythem
template.verticalcheckboxlist
criteria.properties 1.5
global.properties 1.8
>hibernate.cfg.xml 1.65
log4j.properties 1.2
parameters.properties 1.3
struts.properties 1.2
struts.xml 1.12
struts2.xml 1.60
struts3.xml 1.53
struts4.xml 1.11
JRE System Library [JavaSE-1.6]
Web App Libraries
ccMasto

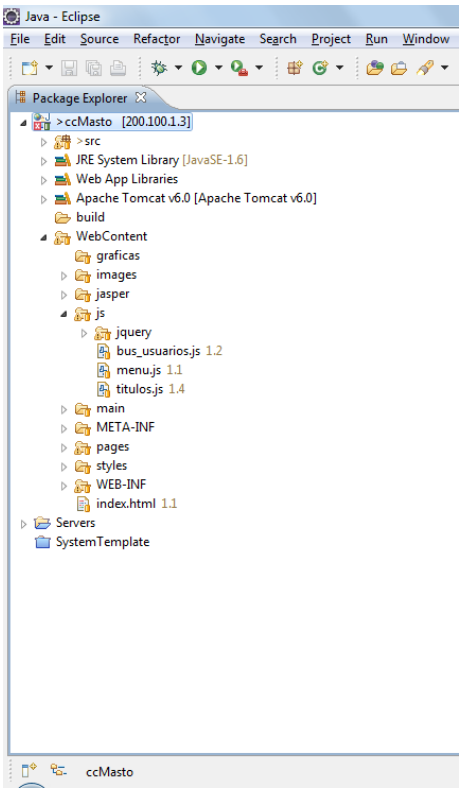

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
MedidaTiempoAction.java 1.4
MenuAction.java 1.7
MessagesAction.java 1.1
ModeloAction.java 1.5
ModoExposicionAction.java 1.4
MonitoreoAction.java 1.7
MonitoreoxEstatusAction.java 1.1
PeliculaPantallaAction.java 1.2
PerfilAction.java 1.2
PosicionDetectorAction.java 1.4
PosicionSelectorAction.java 1.3
ProcesadorPeliculaAction.java 1.3
ProcesadorPeliculaReferenciaAction.java 1.4
ProgramacionAction.java 1.19
ProgramacionCapturaAction.java 1.16
PruebaAction.java 1.9
ReceptorImagenAction.java 1.13
RegistroCalidadImagenAction.java 1.4
RegistroCalidadImagenActualIAction.java 1.5
RetencionFijadorAction.java 1.2
SeccionAction.java 1.3
TipoInstrumentoAction.java 1.4
TipoMastografoAction.java 1.5
TipoProgramacionAction.java 1.3
UnidadMedicaAction.java 1.9
UnidadMedicaDefinePruebaAction.java 1.9
UsersAction.java 1.1
UsuarioAction.java 1.1
UsuarioRegistroAction.java 1.8
UsuarioSessionDatosAction.java 1.3
UtileriasAction.java 1.1
mx.com.solsoft.web.reportes
template.mythem
ccMasto

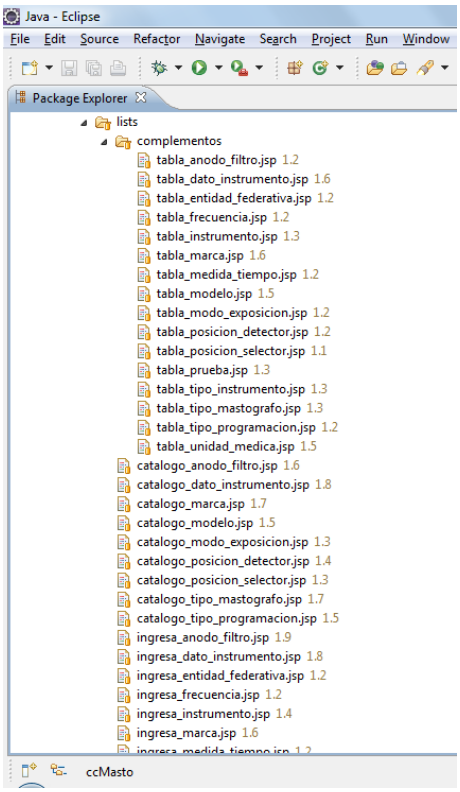

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
mx.com.solsoft.web.reportes
ArtefactoCrReporteAction.java 1.2
ArtefactoDrReporteAction.java 1.3
CalidadImagenReporteAction.java 1.4
ConstanciaGlobalReporteAction.java 1.2
CuartoOscuroReporteAction.java 1.2
EquipamientoInstalacionReporteAction.java 1.2
EstudioRepetidoReporteAction.java 1.3
EvaluaImpresoraInspeccionReporteAction.java 1.1
EvaluaImpresoraNivelHomogeneidadReporteAction.java 1.1
EvaluaImpresoraSensitometriaReporteAction.java 1.2
EvaluaMonitorInspeccionReporteAction.java 1.1
EvaluaMonitorReporteAction.java 1.1
FuerzaCompresionReporteAction.java 1.2
LanzamientoReporteAction.java 1.2
PeliculaPantallaReporteAction.java 1.1
ProcesadorPeliculaReporteAction.java 1.3
ReceptorImagenReporteAction.java 1.2
RegistroCalidadImagenActualReporteAction.java 1.5
ResumenReporteAction.java 1.3
RetencionFijadorReporteAction.java 1.2
template.mythem
template.verticalcheckboxlist
criteria.properties 1.5
global.properties 1.8
hibernate.cfg.xml 1.65
log4j.properties 1.2
parameters.properties 1.3
struts.properties 1.2
struts.xml 1.12
struts2.xml 1.60
struts3.xml 1.53
struts4.xml 1.11
ccMasto

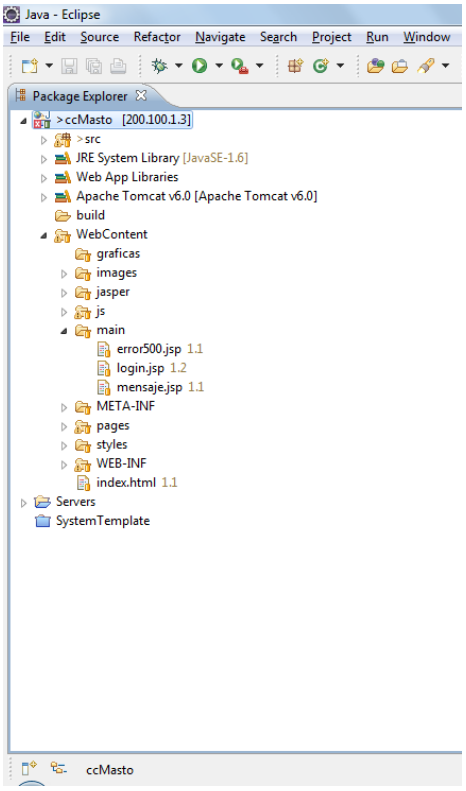

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
jasper
pdf
artefactoCr.jasper 1.2
artefactoCr.jrxml 1.2
artefactoDr.jasper 1.5
artefactoDr.jrxml 1.4
calidadImages.jasper 1.1
calidadImages.jrxml 1.1
constanciaGlobal.jasper 1.3
constanciaGlobal.jrxml 1.2
cuartoOscuro.jasper 1.2
cuartoOscuro.jrxml 1.3
equipamientoInstalacion.jasper 1.2
equipamientoInstalacion.jrxml 1.2
estudiosRepetidos.jasper 1.4
estudiosRepetidos.jrxml 1.5
evaluaMonitor.jasper 1.1
evaluaMonitor.jrxml 1.1
evaluaMonitorInspeccion.jasper 1.3
evaluaMonitorInspeccion.jrxml 1.2
fuerzaCompresion.jasper 1.3
fuerzaCompresion.jrxml 1.4
impresorasImpresion.jasper 1.2
impresorasImpresion.jrxml 1.2
impresorasNivelHomogeneidad.jasper 1.2
impresorasNivelHomogeneidad.jrxml 1.2
impresorasSensitometria.jasper 1.2
impresorasSensitometria.jrxml 1.2
peliculaPantalla.jasper 1.1
peliculaPantalla.jrxml 1.1
procesadorPelicula.jasper 1.2
procesadorPelicula.jrxml 1.2
procesadorPelicula2.jasper 1.3
ccMasto

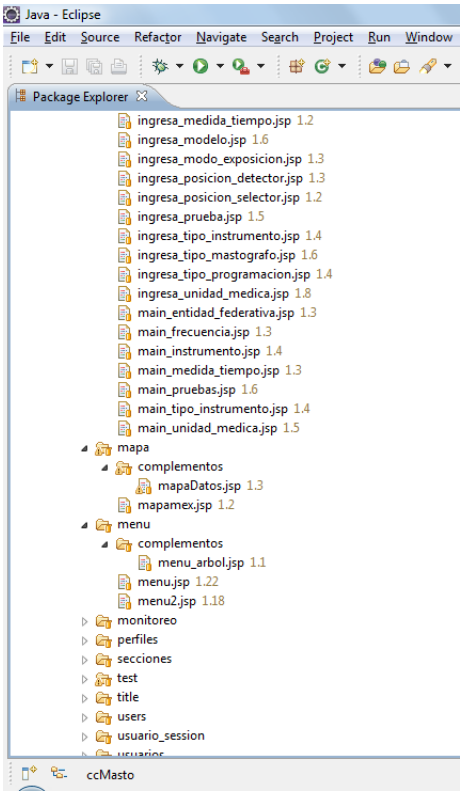

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
ccMasto [200.100.1.3]
src
JRE System Library [JavaSE-1.6]
Web App Libraries
Apache Tomcat v6.0 [Apache Tomcat v6.0]
build
WebContent
graficas
images
jasper
js
main
META-INF
pages
styles
WEB-INF
index.html 1.1
Servers
SystemTemplate
ccMasto

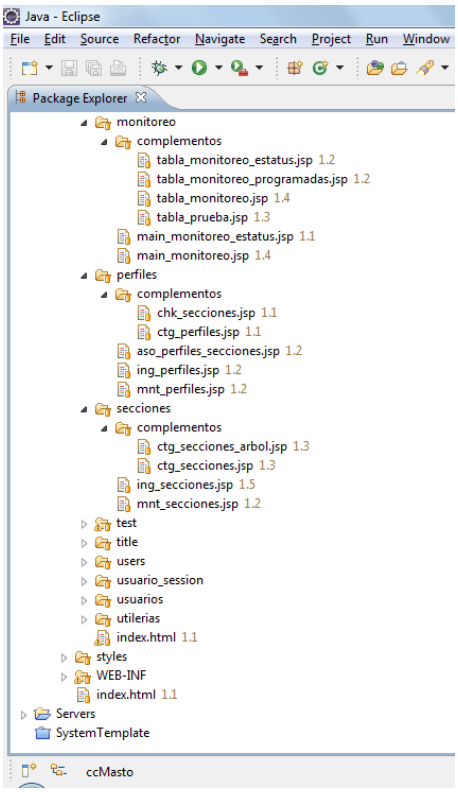

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
fuerzaCompresion.jasper 1.3
fuerzaCompresion.jrxml 1.4
impresorasImpresion.jasper 1.2
impresorasImpresion.jrxml 1.2
impresorasNivelHomogeneidad.jasper 1.2
impresorasNivelHomogeneidad.jrxml 1.2
impresorasSensitometria.jasper 1.2
impresorasSensitometria.jrxml 1.2
peliculaPantalla.jasper 1.1
peliculaPantalla.jrxml 1.1
procesadorPelicula.jasper 1.2
procesadorPelicula.jrxml 1.2
procesadorPelicula2.jasper 1.3
procesadorPelicula2.jrxml 1.3
receptorImages.jasper 1.2
receptorImages.jrxml 1.3
registroCalidadImagenActual.jasper 1.4
registroCalidadImagenActual.jrxml 1.4
registroCalidadImagenActualSub.jasper 1.2
registroCalidadImagenActualSub.jrxml 1.2
resumenControl.jasper 1.4
resumenControl.jrxml 1.4
retencionFijador.jasper 1.3
retencionFijador.jrxml 1.3
js
main
META-INF
pages
styles
WEB-INF
index.html 1.1
Servers
SystemTemplate
ccMasto

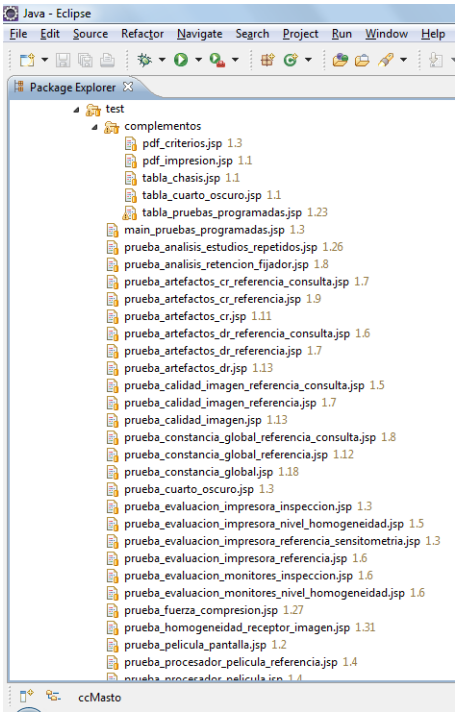

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
>ccMasto [200.100.1.3]
>src
JRE System Library [JavaSE-1.6]
Web App Libraries
Apache Tomcat v6.0 [Apache Tomcat v6.0]
build
WebContent
graficas
images
jasper
js
jquery
bus_usuarios.js 1.2
menu.js 1.1
titulos.js 1.4
main
META-INF
pages
styles
WEB-INF
index.html 1.1
Servers
SystemTemplate
ccMasto

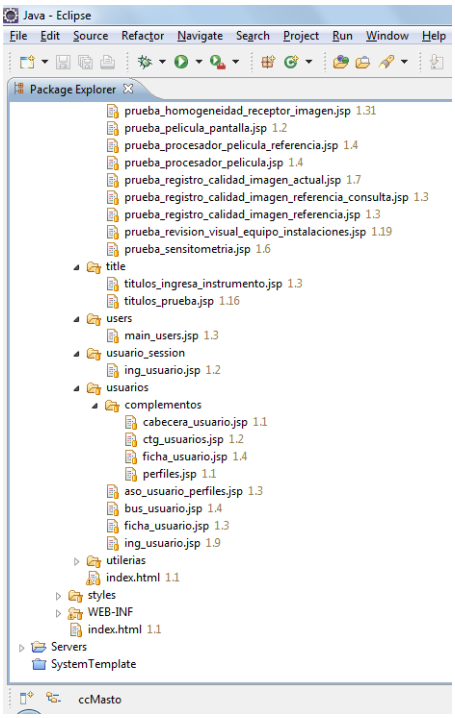

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
lists
complementos
tabla_anodo_filtro.jsp 1.2
tabla_dato_instrumento.jsp 1.6
tabla_entidad_federativa.jsp 1.2
tabla_frecuencia.jsp 1.2
tabla_instrumento.jsp 1.3
tabla_marca.jsp 1.6
tabla_medida_tiempo.jsp 1.2
tabla_modelo.jsp 1.5
tabla_modo_exposicion.jsp 1.2
tabla_posicion_detector.jsp 1.2
tabla_posicion_selector.jsp 1.1
tabla_prueba.jsp 1.3
tabla_tipo_instrumento.jsp 1.3
tabla_tipo_mastografo.jsp 1.3
tabla_tipo_programacion.jsp 1.2
tabla_unidad_medica.jsp 1.5
catalogo_anodo_filtro.jsp 1.6
catalogo_dato_instrumento.jsp 1.8
catalogo_marca.jsp 1.7
catalogo_modelo.jsp 1.5
catalogo_modo_exposicion.jsp 1.3
catalogo_posicion_detector.jsp 1.4
catalogo_posicion_selector.jsp 1.3
catalogo_tipo_mastografo.jsp 1.7
catalogo_tipo_programacion.jsp 1.5
ingresa_anodo_filtro.jsp 1.9
ingresa_dato_instrumento.jsp 1.8
ingresa_entidad_federativa.jsp 1.2
ingresa_frecuencia.jsp 1.2
ingresa_instrumento.jsp 1.4
ingresa_marca.jsp 1.6
ccMasto

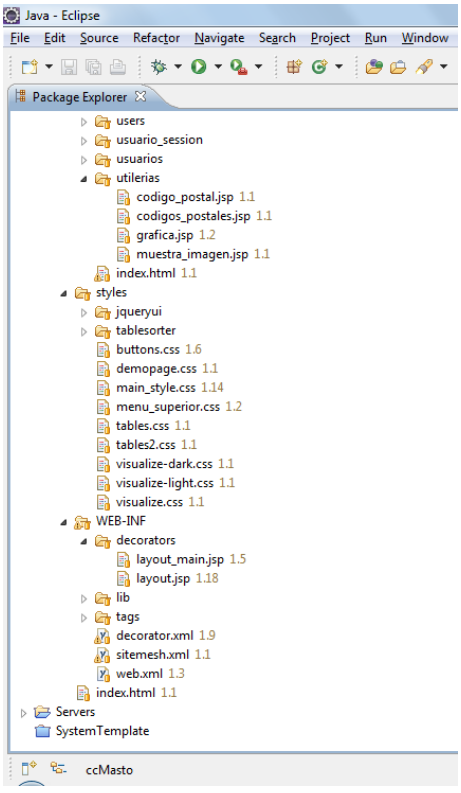

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
>ccMasto [200.100.1.3]
>src
JRE System Library [JavaSE-1.6]
Web App Libraries
Apache Tomcat v6.0 [Apache Tomcat v6.0]
build
WebContent
graficas
images
jasper
js
main
error500.jsp 1.1
login.jsp 1.2
mensaje.jsp 1.1
META-INF
pages
styles
WEB-INF
index.html 1.1
Servers
SystemTemplate
ccMasto


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
ingresa_medida_tiempo.jsp 1.2
ingresa_modelo.jsp 1.6
ingresa_modo_exposicion.jsp 1.3
ingresa_posicion_detector.jsp 1.3
ingresa_posicion_selector.jsp 1.2
ingresa_prueba.jsp 1.5
ingresa_tipo_instrumento.jsp 1.4
ingresa_tipo_mastografo.jsp 1.6
ingresa_tipo_programacion.jsp 1.4
ingresa_unidad_medica.jsp 1.8
main_entidad_federativa.jsp 1.3
main_frecuencia.jsp 1.3
main_instrumento.jsp 1.4
main_medida_tiempo.jsp 1.3
main_pruebas.jsp 1.6
main_tipo_instrumento.jsp 1.4
main_unidad_medica.jsp 1.5
mapa
complementos
mapaDatos.jsp 1.3
mapamex.jsp 1.2
menu
complementos
menu_arbol.jsp 1.1
menu.jsp 1.22
menu2.jsp 1.18
monitoreo
perfiles
secciones
test
title
users
usuario_session
ccMasto


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
monitoreo
complementos
tabla_monitoreo_estatus.jsp 1.2
tabla_monitoreo_programadas.jsp 1.2
tabla_monitoreo.jsp 1.4
tabla_prueba.jsp 1.3
main_monitoreo_estatus.jsp 1.1
main_monitoreo.jsp 1.4
perfiles
complementos
chk_secciones.jsp 1.1
ctg_perfiles.jsp 1.1
aso_perfiles_secciones.jsp 1.2
ing_perfiles.jsp 1.2
mnt_perfiles.jsp 1.2
secciones
complementos
ctg_secciones_arbol.jsp 1.3
ctg_secciones.jsp 1.3
ing_secciones.jsp 1.5
mnt_secciones.jsp 1.2
test
title
users
usuario_session
usuarios
utilerias
index.html 1.1
styles
WEB-INF
index.html 1.1
Servers
SystemTemplate
ccMasto


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
test
complementos
pdf_criterios.jsp 1.3
pdf_impresion.jsp 1.1
tabla_chasis.jsp 1.1
tabla_cuarto_oscuro.jsp 1.1
tabla_pruebas_programadas.jsp 1.23
main_pruebas_programadas.jsp 1.3
prueba_analisis_estudios_repetidos.jsp 1.26
prueba_analisis_retencion_fijador.jsp 1.8
prueba_artefactos_cr_referencia_consulta.jsp 1.7
prueba_artefactos_cr_referencia.jsp 1.9
prueba_artefactos_cr.jsp 1.11
prueba_artefactos_dr_referencia_consulta.jsp 1.6
prueba_artefactos_dr_referencia.jsp 1.7
prueba_artefactos_dr.jsp 1.13
prueba_calidad_imagen_referencia_consulta.jsp 1.5
prueba_calidad_imagen_referencia.jsp 1.7
prueba_calidad_imagen.jsp 1.13
prueba_constancia_global_referencia_consulta.jsp 1.8
prueba_constancia_global_referencia.jsp 1.12
prueba_constancia_global.jsp 1.18
prueba_cuarto_oscuro.jsp 1.3
prueba_evaluacion_impresora_inspeccion.jsp 1.3
prueba_evaluacion_impresora_nivel_homogeneidad.jsp 1.5
prueba_evaluacion_impresora_referencia_sensitometria.jsp 1.3
prueba_evaluacion_impresora_referencia.jsp 1.6
prueba_evaluacion_monitores_inspeccion.jsp 1.6
prueba_evaluacion_monitores_nivel_homogeneidad.jsp 1.6
prueba_fuerza_compresion.jsp 1.27
prueba_homogeneidad_receptor_imagen.jsp 1.31
prueba_pelicula_pantalla.jsp 1.2
prueba_procesador_pelicula_referencia.jsp 1.4
ccMasto


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
prueba_homogeneidad_receptor_imagen.jsp 1.31
prueba_pelicula_pantalla.jsp 1.2
prueba_procesador_pelicula_referencia.jsp 1.4
prueba_procesador_pelicula.jsp 1.4
prueba_registro_calidad_imagen_actual.jsp 1.7
prueba_registro_calidad_imagen_referencia_consulta.jsp 1.3
prueba_registro_calidad_imagen_referencia.jsp 1.3
prueba_revision_visual_equipo_instalaciones.jsp 1.19
prueba_sensitometria.jsp 1.6
title
titulos_ingresa_instrumento.jsp 1.3
titulos_prueba.jsp 1.16
users
main_users.jsp 1.3
usuario_session
ing_usuario.jsp 1.2
usuarios
complementos
cabecera_usuario.jsp 1.1
ctg_usuarios.jsp 1.2
ficha_usuario.jsp 1.4
perfiles.jsp 1.1
aso_usuario_perfiles.jsp 1.3
bus_usuario.jsp 1.4
ficha_usuario.jsp 1.3
ing_usuario.jsp 1.9
utilerias
index.html 1.1
styles
WEB-INF
index.html 1.1
Servers
SystemTemplate
ccMasto


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
users
usuario_session
usuarios
utilerias
codigo_postal.jsp 1.1
codigos_postales.jsp 1.1
grafica.jsp 1.2
muestra_imagen.jsp 1.1
index.html 1.1
styles
jqueryui
tablesorter
buttons.css 1.6
demopage.css 1.1
main_style.css 1.14
menu_superior.css 1.2
tables.css 1.1
tables2.css 1.1
visualize-dark.css 1.1
visualize-light.css 1.1
visualize.css 1.1
WEB-INF
decorators
layout_main.jsp 1.5
layout.jsp 1.18
lib
tags
decorator.xml 1.9
sitemesh.xml 1.1
web.xml 1.3
index.html 1.1
Servers
SystemTemplate
ccMasto

Para comenzar la codificación de los programas, se decidió construir el sistema con *Struts* debido a que este *Framework* nos presenta mejores ventajas en los tiempos de desarrollo, y dado que el usuario tenía planeado realizar una prueba piloto en corto tiempo, se pensó que *Struts* era la mejor opción.

El primer paso en el desarrollo fue definir como se vería el encabezado y el pie de página de la solución de software, y con base al sitio del Centro Nacional de Equidad de Género y Salud Reproductiva, además tomando en cuenta el estilo visual de otros sistemas desarrollados para el CNEGYSR se decidió que el encabezado y el pie de página se vieran de la siguiente forma:

Figura 4.6 ***Encabezado.***

Figura 4.7 ***Pie de Página.***

Al inicio el encabezado era muy sencillo y después se le agregaron elementos como la fecha, hora, nombre del usuario en sesión, perfil del usuario y el menú.

Para llamar al encabezado y al pie de página usamos un *plug-in* incluido en *Struts* llamado *SiteMesh*. El *SiteMesh* es un *framework* que sirve para decorar los sitios web. Se genera un *decorator* (define una plantilla para dar una visualización uniforme a la solución de software). A continuación tenemos un fragmento del *decorator* que se usó para la solución de software:

```
<body>
    <div class="MainDiv">
        <div id="info_pag">
<img alt="" src="images/CNEGySR.png" />
          <div align="left" style="width: 100%" >
               <table width="100%">
                   <tr>
                       <td width="40%" align="left" valign="top">
                           <etiquetas:clock />
                       </td>
                       <td width="43%" align="center"
valign="top" style="color: white;text-transform: uppercase;" >
                           <span >
                           BIENVENIDA(O) <s:property
value="#session.usuarioSesion.nombre"/>
                                         <s:property
value="#session.usuarioSesion.apellidoPaterno"/>
```

```
                                        <s:property
value="#session.usuarioSesion.apellidoMaterno"/>
                                </span>
                            </td>
                            <td valign="top"  width="20%"
align="right"  style="color: white;text-transform: uppercase;">
                                <span title="Estas en el sistema como
<s:property
value='#session.usuarioSesion.perfilSession.nombrePerfil'/>">LOG:
<s:property
value="#session.usuarioSesion.perfilSession.nombreCorto"/></span>
                            </td>
                            <td width="30%" align="right"
valign="top">
                                    <div class="menu_perfil_">
                                    <ul class="nav2"
style="margin:0;padding:0;list-style:none;"  >
                                        <li >
                                            <a title="Modificar Datos"
href="usuarioS" >
                                                <img
src="images/options/avatar.png" />
                                            </a>
                                            <s:if
test="#session.perfilesUsuario.size()!=0" >
                                            <ul>
                                                <s:iterator
value="#session.perfilesUsuario" status="cuenta"  >
                                                    <li>
                                                        <form
method="get" action="login!updatePerfilSession" >
                                                            <input
type="hidden" name="idPerfilSession" value="${idPerfil}" />
                                                            <a
onclick="javascript:$(this).parent().submit()"
class="enlace_perfil" href="#" style="padding:5px
12px;">${nombrePerfil}</a>
                                                        </form>
                                                    </li>
                                                </s:iterator>
                                            </ul>
                                            </s:if>
                                        </li>
                                        <li>
                                             <a title="Desconectarme"
href="login!desconectar"  >
                                                <img
src="images/options/door_out.png" />
                                            </a>
                                        </li>
                                    </ul>
                                    </div>
```

```
                                <s:else>
                                    <a title="Desconectarme"
href="login!desconectar"  >
                                        <img
src="images/options/door_out.png" />
                                    </a>
                                </s:else>
                            </td>
                        </tr>
                    </table>
                </div>
            </div>
        </div>
        <etiquetas:menu_superior />
        <div class="MainDiv">
            <div align="center" style="padding-top: 10px;background-
image: url('images/fondo2.png');width: 977px;" >
                <decorator:body/>
            </div>
        </div>
        <div class="MainDiv" style="padding-top: 10px;">
                <div id="footer_pag" style="color: white;font-
size:13px;font-weight:bolder;padding-top:5px;font-style:italic;">
            Sistema de Control de Calidad en Mastograf
            </div>
        </div>
</body>
```

El *SiteMesh* hace que todas las páginas del sistema se desplieguen usando los elementos del *decorator*. Hay algunas páginas que no deseamos que se le desplieguen las mismas reglas del *decorator*, entonces lo que se hace es decirle al *SiteMesh* que se excluyan. A continuación se muestra el decorator.xml donde se indica que directorios queden excluidos:

```
<?xml version="1.0" encoding="UTF-8"?>
<decorators defaultdir="/WEB-INF/decorators">  <!--donde estan los
decoradores-->
<!-- configuras los layauts en donde No se van a acopar -->
    <excludes> <!--  exluir los que esten bajo a esta carpeta -->
        <pattern>/styles/*</pattern> <!-- css -->
        <pattern>/images/*</pattern>
        <pattern>/index.html</pattern>
        <pattern>/pages/lists/complementos/*</pattern>
        <pattern>/pages/gestion/complementos/*</pattern>
        <pattern>/pages/monitoreo/complementos/*</pattern>
        <pattern>/pages/depending-combo/*</pattern>
Fvko.m, 84583
        <pattern>/pages/usuarios/complementos/*</pattern>
        <pattern>/pages/secciones/complementos/*</pattern>
        <pattern>/pages/perfiles/complementos/*</pattern>
```

```
        <pattern>/pages/menu/complementos/*</pattern>
        <pattern>/pages/combos/*</pattern>
        <pattern>/pages/utilerias/*</pattern>
        <pattern>/pages/mapa/complementos/*</pattern>
        <pattern>/pages/title/*</pattern>
           <pattern>/js/*</pattern>
    </excludes>
    <decorator name="layout" page="layout.jsp">
        <pattern>/pages/*</pattern>
    </decorator>
    <decorator name="layout_main" page="layout_main.jsp">
        <pattern>/main/*</pattern>
    </decorator>
</decorators>
```

Lo siguiente que se hizo fue definir la hoja de estilo que se usó para el desarrollo de la solución de software. De igual forma la hoja de estilo se basó en los sistemas anteriores del CNEGYSR. A continuación tenemos la hoja de estilos principal:

```
.MainDiv{
      width: 1012px;
      text-align: center;
      margin: auto;
}
img {
      border: none;
}
#info_pag {
      background-color: gray;
      overflow:hidden;
      width: 975px;
      height: 106px;
      margin-left: 3px;
      margin-top: -10px;
}
#footer_pag {
      background-color: gray;
      overflow:hidden;
      width: 975px;
      height: 24px;
      margin-left: 3px;
}
#menu {
      overflow:hidden;
      width: 975px;
      height: 20px;
      margin-left: 3px;
      text-align: left;
}
#main_pag {
      background-color: #CACFCE;
      position: fixed;
      overflow: auto;
       height: 100%;
       width: 150px;
       margin-left: 3px;
}
/** Titulos de las pantallas */
.titles {
      text-transform: uppercase;
      font-size: 12px;
      font-weight: bolder;
      color:  rgba(114, 115, 140,
1) ;
      padding-bottom: 5px;
}
/** Opciones de las pantallas */
.options {
      width: 90%;
      font-size: 9pt;
      color:  rgba(114, 115, 140,
1) ;
      text-align: left;
}
.options img{
      height: 12px;
      width: 12px;
      cursor: pointer;
}
.options a{
      text-decoration: none;
      color:  rgba(114, 115, 140,
1) ;
}
.options input[type='text'].buscar{
```

```
	height: 11px;
	width: 200px;
	font-size: 10px;
}
input{
	background-color: rgba(199,
209, 217, 0.3) ;
}
select{
	background-color: rgba(199,
209, 217, 0.3) ;
	font-size: 11px;
}
input:FOCUS {
	background-color: WHITE ;
}
select:FOCUS {
	background-color: rgba(199,
209, 217, 0.3) ;
}
.form {
	width: 90%;
	font-size: 9pt;
	text-align: left;
	padding: 2px;
	border: .5px double #000;
	background-color: #DEDEDE ;
}

.Menu {
	padding: 10px;
	width: 90%;
	text-align: left;
	background-color: rgba(199,
209, 217, 0.3) ;
}
.Menu a{
	text-decoration: none;
}
.clock {
	color: white;
	font-size: 12px;
	font-style: italic;
	vertical-align: top;
	text-align: left;
}
.botonL{
	background-color: #EDEEF2 ;

}
.botonL:FOCUS {
	background-color: white ;
}
.botonReadOnly{
	background-color: #EDEEF2 ;
	border:none;
}
.botonReadOnly:FOCUS {
	background-color: #EDEEF2 ;
}
.titlesGeneral{
	font-size: 8pt;
	color:#000000;
	font-weigth:800;
	font-style:normal;
	font-weight: bold;
	font-family: arial;
}
.titlesGeneral2{
	font-size: 8pt;
	color:#000000;
	font-weigth:800;
	font-style:italic;
	font-weight: bold;
	font-family: arial;
}
.titlesGeneral3{
	font-size: 14px;
	color:#59464D;
	font-weigth:800;
	font-style:italic;
	font-weight: normal;
	font-family: arial;
}
.titlesGeneral4{
	font-size: 7pt;
	color:#000000;
	font-weigth:800;
	font-style:normal;
	font-weight: bold;
	font-family: arial;
}
.titlesGeneral5{
	font-size: 9pt;
	color:#000000;
	font-weigth:800;
	font-style:normal;
	font-weight: normal;
	font-family: arial;
}
.titlesGeneral6{
	font-size: 9pt;
	color:#000000;
	font-weigth:800;
	font-style:normal;
	font-weight: normal;
	font-family: arial;
	text-transform: lowercase;
	text-transform: uppercase;
	text-transform: none;
}
.botonVmpGeneral{
	background-color: #EDEEF2 ;
```

```
}
.botonVmpGeneral:FOCUS {
      background-color: white ;
}
.botonVmpCentral{
      background-color: #EDEEF2 ;
}
.botonVmpCentral:FOCUS {
      background-color: white ;
}
.botonVmpCentral1{
      background-color: #EDEEF2 ;
}
.botonVmpCentral1:FOCUS {
      background-color: white ;
}
.botonVmpGeneral1{
      background-color: #EDEEF2 ;
}
.botonVmpGeneral1:FOCUS {
      background-color: white ;
}
.botonVmpCentral2{
      background-color: #EDEEF2 ;

}
.botonVmpCentral2:FOCUS {
      background-color: white ;
}
.botonVmpCentral3{
      background-color: #EDEEF2 ;

}
.botonVmpCentral3:FOCUS {
      background-color: white ;
}
.superIndice {
      vertical-align: top;
      font-size:7px;
}
.subIndice {
      vertical-align: bottom;
      font-size:7px;s
}
.normally {
      vertical-align: middle;
      font-size:8px;s
}
.botonReadOnlyTwo{
      background-color: rgba(178,
209, 224,0.8);
      border:none;
}
.botonReadOnlyTwo:FOCUS {
      background-color: rgba(178,
209, 224,0.8);
}
.botonReadOnlyThree{
      background-color: rgba(199,
209, 217, 0.3);
      border:none;
}
.botonReadOnlyThree:FOCUS {
      background-color: rgba(199,
209, 217, 0.3) ;
}
```

Una vez que la parte visual quedó definida, entonces comenzó el desarrollo de la solución del software.

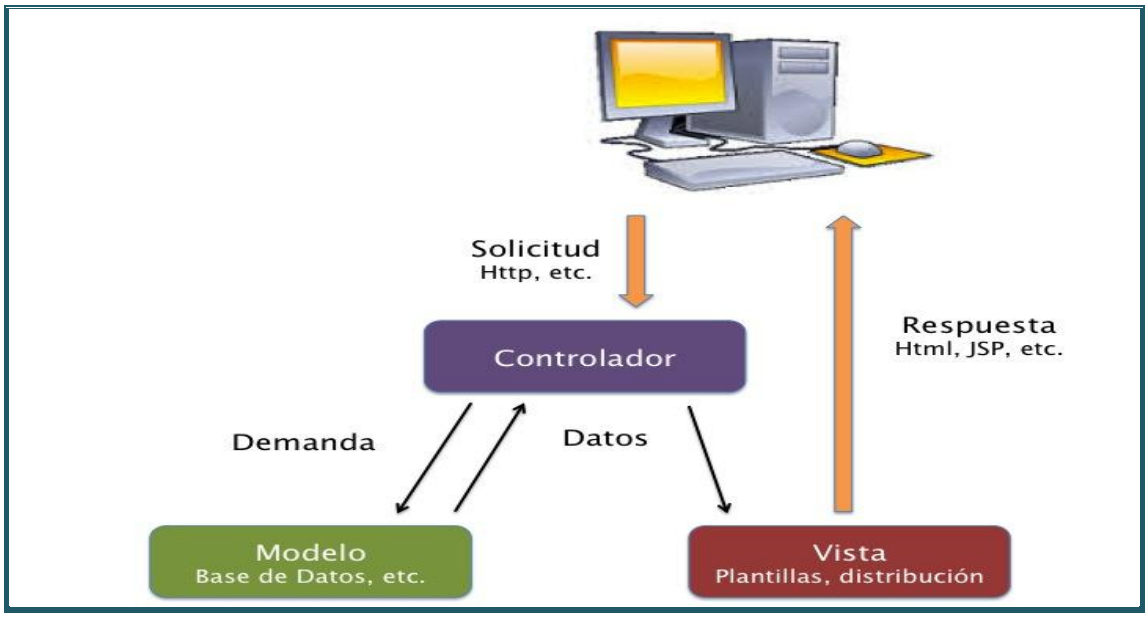


Figura 4.5 ***Diagrama de MVC.***

Struts utiliza el Modelo-Vista-Controlador, por lo que todas las pantallas creadas para la solución de software cuentan con:

a) **Modelo** - Está conformado por todas las clases que se crearon para acceder a las tablas de la base de datos (usando *hibernate*). Estas clases son conocidas como entidades, y están conformadas por componentes tipo JavaBeans, así como con elementos propios del *hibernate*. El paquete que usamos por default es el ***mx.com.solsoft.entities***. A continuación se muestra un ejemplo de una entidad:

```
package mx.com.solsoft.entities;
import java.io.Serializable;
import java.util.List;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table (name="instrumento")
public class Instrumento implements Serializable{
     private static final long serialVersionUID = 1L;
     private int idInstrumento;
     private String nombreInstrumento;
```

```
	private boolean estatusInstrumento = true;
	private TipoInstrumento tipoInstrumento;
	private List<DatoInstrumento> datoInstrumento;
	private List<DefinePruebaInstrumento> definePruebaInstrumentos;
	private boolean requiereTipoMastografo = false;
	private boolean estacionInstrumento = false;

	@Id
	@GeneratedValue
	@Column(name="id_instrumento")
	public int getIdInstrumento() {
		return idInstrumento;
	}
	public void setIdInstrumento(int idInstrumento) {
		this.idInstrumento = idInstrumento;
	}
	@Column(name="nombre_instrumento")
	public String getNombreInstrumento() {
		return nombreInstrumento;
	}
	public void setNombreInstrumento(String nombreInstrumento) {
		this.nombreInstrumento = nombreInstrumento;
	}
	@Column(name="estatus_instrumento")
	public boolean getEstatusInstrumento() {
		return estatusInstrumento;
	}
	public void setEstatusInstrumento(boolean estatusInstrumento) {
		this.estatusInstrumento = estatusInstrumento;
	}
	@ManyToOne
	@JoinColumn(name="id_tipo_instrumento")
	public TipoInstrumento getTipoInstrumento() {
		return tipoInstrumento;
	}
	public void setTipoInstrumento(TipoInstrumento tipoInstrumento) {
		this.tipoInstrumento = tipoInstrumento;
	}
	@OneToMany
	@JoinColumn(name="id_dato_instrumento",
updatable=false,insertable=false)	public List<DatoInstrumento>
getDatoInstrumento() {
		return datoInstrumento;
	}
	public void setDatoInstrumento(List<DatoInstrumento> datoInstrumento)
{
		this.datoInstrumento = datoInstrumento;
	}
	@OneToMany
	@JoinColumn (name="id_instrumento", updatable=false,insertable=false)
	public List<DefinePruebaInstrumento> getDefinePruebaInstrumentos() {
		return definePruebaInstrumentos;
```

```
	}
	public void setDefinePruebaInstrumentos(List<DefinePruebaInstrumento>
definePruebaInstrumentos) {
		this.definePruebaInstrumentos = definePruebaInstrumentos;
	}
	@Column(name="requiere_tipo_mastografo_instrumento")
	public boolean getRequiereTipoMastografo() {
		return requiereTipoMastografo;
	}
	public void setRequiereTipoMastografo(boolean requiereTipoMastografo)
{
		this.requiereTipoMastografo = requiereTipoMastografo;
	}
	@Column(name="estacion_instrumento")
	public boolean isEstacionInstrumento() {
		return estacionInstrumento;
	}
	public void setEstacionInstrumento(boolean estacionInstrumento) {
		this.estacionInstrumento = estacionInstrumento;
	}
```

Clase Instrumento (Entidad)

Una vez que se definió la entidad, se necesita indicar al hibernate que dicha clase nos permite acceder a la tabla de la base de datos, y esto se declara en el archivo hibernate.xml, a continuación ponemos un fragmento del mismo:

```
	<!– Entidades que serán mapeadas con la base de datos ccmasto -->
	 <mapping class="mx.com.solsoft.entities.Usuario"/>
	 <mapping class="mx.com.solsoft.entities.Marca" />
	 <mapping class="mx.com.solsoft.entities.Prueba" />
	 <mapping class="mx.com.solsoft.entities.EntidadFederativa" />
	 <mapping class="mx.com.solsoft.entities.TipoMastografo" />
	 <mapping class="mx.com.solsoft.entities.UnidadMedica" />
	 <mapping class="mx.com.solsoft.entities.TipoInstrumento"/>
	 <mapping class="mx.com.solsoft.entities.MedidaTiempo"/>
	 <mapping class="mx.com.solsoft.entities.Frecuencia"/>
	 <mapping class="mx.com.solsoft.entities.AnodoFiltro"/>
	 <mapping class="mx.com.solsoft.entities.AnodoFiltroTipoMastografo"/>
	 <mapping class="mx.com.solsoft.entities.Modelo"/>
	 <mapping class="mx.com.solsoft.entities.Frecuencia"/>
	 <mapping class="mx.com.solsoft.entities.Instrumento"/>
	 <mapping class="mx.com.solsoft.entities.MarcaTipoInstrumento"/>
	 <mapping class="mx.com.solsoft.entities.DatoInstrumento"/>
	 <mapping class="mx.com.solsoft.entities.DefinePrueba"/>
	 <mapping class="mx.com.solsoft.entities.TipoProgramacion"/>
	 <mapping class="mx.com.solsoft.entities.ProgramacionPrueba"/>
	 <mapping class="mx.com.solsoft.entities.FuerzaCompresion"/>
	 <mapping
class="mx.com.solsoft.entities.FuerzaCompresionInstrumento"/>
	 <mapping class="mx.com.solsoft.entities.UnidadMedicaDefinePrueba"/>
	 <mapping class="mx.com.solsoft.entities.DefinePruebaInstrumento"/>
```

```
      <mapping class="mx.com.solsoft.entities.EquipamientoInstalacion"/>
      <mapping
class="mx.com.solsoft.entities.EquipamientoInstalacionInstrumento"/>
      <mapping class="mx.com.solsoft.entities.EstudioRepetido"/>
      <mapping class="mx.com.solsoft.entities.EstudioRepetidoInstrumento"/>
      <mapping class="mx.com.solsoft.entities.ModoExposicion"/>
      <mapping class="mx.com.solsoft.entities.PosicionDetector"/>
       <mapping class="mx.com.solsoft.entities.PosicionSelector"/>
      <mapping class="mx.com.solsoft.entities.ReceptorImagen"/>
      <mapping class="mx.com.solsoft.entities.ReceptorImagenInstrumento"/>
      <mapping class="mx.com.solsoft.entities.ArtefactoDrReferencia"/>
      <mapping
class="mx.com.solsoft.entities.ArtefactoDrReferenciaInstrumento"/>
      <mapping class="mx.com.solsoft.entities.ArtefactoDr"/>
```

Fragmento de hibernate.xml

También como parte del modelo se tienen las interfaces de las reglas de negocio, así como la implementación de las mismas. Estas reglas de negocio consisten básicamente en todas las consultas, actualizaciones y modificaciones a las tablas de la base de datos usando objetos de las entidades mencionadas. El paquete de las interfaces es **mx.com.solsoft.dao** y el paquete para las implementaciones es **mx.solsoft.daoImp**. A continuación mostramos un ejemplo de una interfaz, así como su implementación:

```
package mx.com.solsoft.dao;
import java.util.List;
import mx.com.solsoft.entities.Instrumento;

public interface InstrumentoDao {
    public List<Instrumento> getAll();
    public Instrumento getById(int idInstrumento);
    public List<Instrumento> getByNombre(String nombreParcial);
    public List<Instrumento> getByStatus(boolean estatusInstrumento);
    public List<Instrumento> getByTipoInstrumento(String
nombreParcialTipoInstrumento);
    public List<Instrumento> getByTipoInstrumento(int idTipoInstrumento);
    public void update(Instrumento instrumento);
    public void add(Instrumento instrumento);
    public boolean searchByNombre(String nombreInstrumento);


    /******** METODOS PARA LA OBTENER DATOS DE LA CLASE EMBEBIDA *********/
    public List<Instrumento> getByDefinePrueba(int idDefinePrueba);
```

Interface InstrumentoDao.java

```
package mx.com.solsoft.daoImp;
import java.util.List;
import org.hibernate.Criteria;
```

```
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.criterion.Order;
import org.hibernate.criterion.Restrictions;
import mx.com.solsoft.dao.InstrumentoDao;
import mx.com.solsoft.entities.DefinePruebaInstrumento;
import mx.com.solsoft.entities.Instrumento;
import mx.com.solsoft.utilities.HibernateUtil;
import mx.com.solsoft.utilities.Utilities;

public class InstrumentoDaoImp implements InstrumentoDao {
	@SuppressWarnings("unchecked")
	@Override
	public List<Instrumento> getAll() {
		List<Instrumento> instrumentos = null;
		Session session =null;
		session = HibernateUtil.getSessionFactory().openSession();
		try {
			Criteria criteria =
session.createCriteria(Instrumento.class);
			instrumentos = criteria.list();
		}catch (HibernateException e){
			e.printStackTrace();
		}finally{
			if(session != null) session.close();
		}
		return instrumentos;
	}
	@Override
	public Instrumento getById(int idInstrumento) {
		Instrumento instrumento = null;
		Session session =null;
		session = HibernateUtil.getSessionFactory().openSession();
		try {
			Criteria criteria =
session.createCriteria(Instrumento.class);
			criteria.add(Restrictions.eq("idInstrumento",
idInstrumento));
			instrumento = (Instrumento) criteria.uniqueResult();
		} catch (HibernateException e) {
			e.printStackTrace();
		} finally {
			if (session != null) session.close();
		}
		return instrumento;
	}
	@SuppressWarnings("unchecked")
	@Override
	public List<Instrumento> getByNombre(String nombreParcial) {
		List<Instrumento> instrumentos = null;
		Session session =null;
```

```
            session = HibernateUtil.getSessionFactory().openSession();
            try {
                  Criteria criteria =
session.createCriteria(Instrumento.class);
                  criteria.add(Restrictions.like("nombreInstrumento", "%" +
nombreParcial + "%"));
                  criteria.addOrder(Order.asc("nombreInstrumento"));
//Ordenamiento ascendente
                  instrumentos = criteria.list();
            } catch (HibernateException e) {
                  e.printStackTrace();
            } finally {
                  if (session != null) session.close();
            }
            return instrumentos;
      }
      @SuppressWarnings("unchecked")
      @Override
      public List<Instrumento> getByStatus(boolean estatusInstrumento) {
            List<Instrumento> instrumentos = null;
            Session session =null;
            session = HibernateUtil.getSessionFactory().openSession();
            try {
                  Criteria criteria =
session.createCriteria(Instrumento.class);
                  criteria.add(Restrictions.eq("estatusInstrumento",
estatusInstrumento));
                  criteria.addOrder(Order.asc("nombreInstrumento"));
//Ordenamiento ascendente
                  instrumentos = criteria.list();
            } catch (HibernateException e) {
                  e.printStackTrace();
            } finally {
                  if (session != null) session.close();
            }
            return instrumentos;
      }
      @SuppressWarnings("unchecked")
      @Override
      public List<Instrumento> getByTipoInstrumento(String
nombreParcialTipoInstrumento) {
            List<Instrumento> instrumentos = null;
            Session session =null;
            session = HibernateUtil.getSessionFactory().openSession();
            try {
                  Criteria criteria =
session.createCriteria(Instrumento.class).addOrder(Order.asc("nombreInstrum
ento")).createCriteria("tipoInstrumento").add(Restrictions.like("nombreTipo
Instrumento", "%" + nombreParcialTipoInstrumento + "%"));
                  instrumentos = criteria.list();
            } catch (HibernateException e) {
                  e.printStackTrace();
```

```
            } finally {
                  if (session != null) session.close();
            }
            return instrumentos;
      }
      @SuppressWarnings("unchecked")
      @Override
      public List<Instrumento> getByTipoInstrumento(int idTipoInstrumento)
{
            List<Instrumento> instrumentos = null;
            Session session =null;
            session = HibernateUtil.getSessionFactory().openSession();
            try {
                  Criteria criteria =
session.createCriteria(Instrumento.class);
                  criteria.addOrder(Order.asc("nombreInstrumento"));
//Ordenamiento ascendente
      criteria.add(Restrictions.eq("tipoInstrumento.idTipoInstrumento",
idTipoInstrumento));
                  instrumentos = criteria.list();
            } catch (HibernateException e) {
                  e.printStackTrace();
            } finally {
                  if (session != null) session.close();
            }
            return instrumentos;
      }
      @Override
      public void update(Instrumento instrumento) {
            Session session;
            Transaction transaction = null;
            Utilities.print("SessionFactory...");
            session = HibernateUtil.getSessionFactory().openSession();
            try {
                  transaction = session.beginTransaction();
                  Utilities.print("Actualizando un instrumento...");
                  session.update(instrumento);
                  Utilities.print("El id actualizado fue: " +
instrumento.getIdInstrumento());
                  transaction.commit();
            } catch (HibernateException e) {
                  transaction.rollback();
                  e.printStackTrace();
            } finally {
                  if (session != null) session.close();
            }
      }
      @Override
      public void add(Instrumento instrumento) {
            Session session;
            Transaction transaction = null;
            Utilities.print("SessionFactory...");
```

```
            session = HibernateUtil.getSessionFactory().openSession();
            try {
                  transaction = session.beginTransaction();
                  Utilities.print("Insertando un nuevo instrumento...");
                  session.save(instrumento);
                  Utilities.print("El id generado fue: " +
instrumento.getIdInstrumento());
                  transaction.commit();
            } catch (HibernateException e) {
                  transaction.rollback();
                  e.printStackTrace();
            } finally {
                  if (session != null) session.close();
            }
      }
      @Override
      public boolean searchByNombre(String nombreInstrumento) {
            Instrumento instrumento = null;
            Session session =null;
            boolean existe = false;
            session = HibernateUtil.getSessionFactory().openSession();
            try {
                  Criteria criteria =
session.createCriteria(Instrumento.class);
                  criteria.add(Restrictions.eq("nombreInstrumento",
nombreInstrumento));
                  instrumento = (Instrumento)criteria.uniqueResult();
                  if (instrumento != null && instrumento.getIdInstrumento() >
0)
                        existe = true;
                  else
                        existe = false;
            } catch (HibernateException e) {
                  e.printStackTrace();
            } finally {
                  if (session != null) session.close();
            }
            return existe;
      }
      @SuppressWarnings("unchecked")
      @Override
      public List<Instrumento> getByDefinePrueba(int idDefinePrueba) {
            List<Instrumento> instrumentos = null;
            Session session =null;

            session = HibernateUtil.getSessionFactory().openSession();
            try {
                  Criteria criteria =
session.createCriteria(DefinePruebaInstrumento.class);
                  criteria.add(Restrictions.eq("idDefinePreuba",
idDefinePrueba));
                  instrumentos = criteria.list();
```

```
            } catch (HibernateException e) {
                  e.printStackTrace();
            } finally {
                  if (session != null) session.close();
            }
            return instrumentos;
      }
```

Implementación de Clase InstrumentoDaoImp.java

b) **Controlador** – La función del controlador es administrar las acciones y el flujo del sistema. En Struts, el flujo del sistema está dado por el archivo struts.xml. En nuestro caso, para poder desarrollar al mismo tiempo, cada desarrollador tenía su versión del archivo xml y lo que se hizo fue incluirlos en el struts.xml principal. A continuación tenemos el struts.xml, así como un fragmento de uno de los archivos xml utilizado para el desarrollo del sistema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<struts>
      <constant name="struts.devMode" value="true" />
      <constant name="struts.multipart.maxSize" value="2147483648" />
      <package namespace="uno" name="mx.com.solsoft.web"  extends="struts-
default" >
      <interceptors>
      <interceptor name="validateInteceptor"
class="mx.com.solsoft.interceptors.ValidateInterceptors"></interceptor>
      </interceptors>
            <!-- ********************** GENERAL ***************************
-->
            <global-results>
                  <result name="invalido">/main/login.jsp</result>
                  <result name="menu">/pages/menu/menu.jsp</result>
            </global-results>
            <!-- ******************LOGIN********************** -->
            <!-- action name="login" nombre de la direccion
                        class="mx.com.solsoft.web.LoginAction"  se ejecuta
estas acciones
                        si requiero mas  formatos o .jsp se  declaran  todo
el contenido de las etiquetas acciones -->
            <action name="login" method="execute"
class="mx.com.solsoft.web.LoginAction" >
                  <result>/main/login.jsp</result>
                  <result name="valido">/pages/menu/menu.jsp</result>
                  <result name="input">/main/login.jsp</result>
            </action>
            <action name="menu" method="execute"
class="mx.com.solsoft.web.MenuAction" >
                  <interceptor-ref name="validateInteceptor"/>
                  <interceptor-ref name="defaultStack"/>
                  <result>/pages/menu/menu.jsp</result>
```

```
                <result
name="muestraMenu">/pages/menu/complementos/menu_arbol.jsp</result>
            </action>
      </package>
      <include file="struts2.xml" />
      <include file="struts3.xml" />
      <include file="struts4.xml" />

</struts>
```

Archivo struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<struts>
      <constant name="struts.devMode" value="true" />
      <constant name="struts.multipart.maxSize" value="2147483648" />
      <package name="dos"  extends="mx.com.solsoft.web" >
            <interceptors>
            <interceptor name="validateInteceptor"
class="mx.com.solsoft.interceptors.ValidateInterceptors"></interceptor>
            </interceptors>
<!-- ******************    CATALOGO DE INSTRUMENTO
*********************    -->
            <action name="instrumento" method="execute"
class="mx.com.solsoft.web.InstrumentoAction" >
            <interceptor-ref name="validateInteceptor"/>
            <interceptor-ref name="token" >
                  <param name="includeMethods">guarda</param>
            </interceptor-ref>
            <interceptor-ref name="defaultStack"/>
                  <result>/pages/lists/main_instrumento.jsp</result>
                  <result name="invalido">/main/login.jsp</result>
            <result
name="muestra_tabla">/pages/lists/complementos/tabla_instrumento.jsp</resul
t>
                  <result
name="ingresa">/pages/lists/ingresa_instrumento.jsp</result>
                  <result
name="input">/pages/lists/ingresa_instrumento.jsp</result>
                  <result
name="invalid.token">/pages/lists/main_instrumento.jsp</result>
                  <result name="salvado"
type="redirectAction">instrumento</result>
            </action>
```

Fragmento de struts2.xml

En estos xml anteriores se definen las clases que van a ser las acciones que controlan a nuestra vista. En el archivo struts2.xml se muestra la acción que controla al catálogo de instrumentos asociado con los ejemplos anteriores en la parte del modelo. Podemos ver que la acción define la clase que se va a usar, así como las vistas se van a mandar llamar una vez que se ejecute el evento asociado al atributo name="" de la etiqueta

result. Las clases de acción se están definiendo en el paquete **mx.solsoft.com.web**. A continuación se muestra un ejemplo de una clase de acción:

```
package mx.com.solsoft.web;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import mx.com.solsoft.daoImp.InstrumentoDaoImp;
import mx.com.solsoft.daoImp.TipoInstrumentoDaoImp;
import mx.com.solsoft.entities.Instrumento;
import mx.com.solsoft.entities.TipoInstrumento;
import mx.com.solsoft.utilities.Utilities;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class InstrumentoAction extends ActionSupport implements
ModelDriven<Instrumento> {
	private static final long serialVersionUID = 1L;
	List<Instrumento> instrumentos;
	Instrumento instrumento = new Instrumento();
	private String busqueda;
	private List<TipoInstrumento> tiposInstrumentos;
	private int idTipoInstrumento;
	private boolean equipoTipoInstrumento;
	public String execute(){
		return SUCCESS;
	}
	public String consulta() {
		setInstrumentos(obtieneListaBusqueda());
		return "muestra_tabla";
	}
	public String ingresa() {
		TipoInstrumentoDaoImp tipoInstrumentoDao =  new
TipoInstrumentoDaoImp();
		setTiposInstrumentos(tipoInstrumentoDao.getByEstatus(true));
		return "ingresa";
	}
	public String guarda(){
		InstrumentoDaoImp instrumentoDao = new InstrumentoDaoImp();
		TipoInstrumentoDaoImp tipoInstrumentoDao =  new
TipoInstrumentoDaoImp();
		TipoInstrumento tipoInstrumento =
tipoInstrumentoDao.getById(idTipoInstrumento);
		instrumento.setTipoInstrumento(tipoInstrumento);
		if (instrumento.getIdInstrumento() == 0) {
			instrumentoDao.add(instrumento);
		}else{
			instrumentoDao.update(instrumento);
		}
		return "salvado";
	}
```

```
	public void validateGuarda(){
		//Validamos el nombre del instrumento no venga vacio
		if (instrumento.getNombreInstrumento().trim().equals("")){
			TipoInstrumentoDaoImp tipoInstrumentoDao =  new
TipoInstrumentoDaoImp();

	setTiposInstrumentos(tipoInstrumentoDao.getByEstatus(true));
			addFieldError("nombreInstrumento", "El nombre del
instrumento no puede ir en blanco");
		}
		//Validamos el tamaño adecuado del nombre del instrumento
		if(instrumento.getNombreInstrumento().length() > 30){
			TipoInstrumentoDaoImp tipoInstrumentoDao =  new
TipoInstrumentoDaoImp();

	setTiposInstrumentos(tipoInstrumentoDao.getByEstatus(true));
			addFieldError("nombreInstrumento", "El nombre del
instrumento no puede tener más de 30 caracteres");
		}
		//Validamos que se haya seleccionado un tipo de instrumento
		if(idTipoInstrumento == 0 ){
			TipoInstrumentoDaoImp tipoInstrumentoDao =  new
TipoInstrumentoDaoImp();

	setTiposInstrumentos(tipoInstrumentoDao.getByEstatus(true));
			addFieldError("idTipoInstrumento", "Favor de seleccionar un
tipo de instrumento");
		}
		 //Validamos que no se haya dado de alta con anterioridad el
nombre
		 InstrumentoDaoImp instrumentoDaoImp = new InstrumentoDaoImp();
		 Utilities.print("Id Instrumento: " +
instrumento.getIdInstrumento());
		 if (instrumento.getIdInstrumento() == 0 &&
instrumentoDaoImp.searchByNombre(instrumento.getNombreInstrumento())){
				TipoInstrumentoDaoImp tipoInstrumentoDao =  new
TipoInstrumentoDaoImp();
	setTiposInstrumentos(tipoInstrumentoDao.getByEstatus(true));
				addFieldError("nombreInstrumento", "Ese instrumento
ya existe en la base de datos");
		 }
	}
	public List<Instrumento> obtieneListaBusqueda(){
		List<Instrumento> listaxNombre;
		List<Instrumento> listaxTipoInstrumento = new
ArrayList<Instrumento>();
		List<Instrumento> listaResultado = new ArrayList<Instrumento>();
		boolean update = true;
		String busqueda = getBusqueda();
		InstrumentoDaoImp instrumentoDao = new InstrumentoDaoImp();
		//Obtenemos el listado por nombre
		listaxNombre = instrumentoDao.getByNombre(busqueda);
```

```
            if (!busqueda.trim().equals("")) {
                  //Obtenemos el resultado por Tipo Instrumento
                  listaxTipoInstrumento =
instrumentoDao.getByTipoInstrumento(busqueda);
            }
            //Juntamos las listas en una sola lista de resultados
            update = true;
            if (listaxNombre != null && listaxNombre.size() > 0) {
                  for(Instrumento instrumentoxNombre : listaxNombre){
                        for(Instrumento res: listaResultado) {
                              if (instrumentoxNombre.getIdInstrumento() ==
res.getIdInstrumento()){
                                    update = false;
                                    break;
                              }
                        }
                        if (update)
                              listaResultado.add(instrumentoxNombre);
                  }
            }
            update = true;
            if (listaxTipoInstrumento != null &&
listaxTipoInstrumento.size() > 0) {
                  for (Instrumento unidadesxTipoInstrumento :
listaxTipoInstrumento){
                        for(Instrumento res: listaResultado) {
                              if (unidadesxTipoInstrumento.getIdInstrumento()
== res.getIdInstrumento()){
                                    update = false;
                                    break;
                              }
                        }
                        if (update)
                              listaResultado.add(unidadesxTipoInstrumento);
                  }
            }
            return listaResultado;
      }
      @Override
      public Instrumento getModel() {

            return instrumento;
      }
      public List<Instrumento> getInstrumentos() {
            return instrumentos;
      }
      public void setInstrumentos(List<Instrumento> instrumentos) {
            this.instrumentos = instrumentos;
      }
      public Instrumento getInstrumento() {
            return instrumento;
      }
```

```
	public void setInstrumento(Instrumento instrumento) {
		this.instrumento = instrumento;
	}
	public String getBusqueda() {
		try {
			busqueda = new String(busqueda.getBytes("ISO-8859-1"),
"UTF-8");
		} catch (UnsupportedEncodingException e) {

			e.printStackTrace();
		}
		return busqueda;
	}
	public void setBusqueda(String busqueda) {
		this.busqueda = busqueda;
	}
	public List<TipoInstrumento> getTiposInstrumentos() {
		return tiposInstrumentos;
	}
	public void setTiposInstrumentos(List<TipoInstrumento>
tiposInstrumentos) {
		this.tiposInstrumentos = tiposInstrumentos;
	}
	public int getIdTipoInstrumento() {
		return idTipoInstrumento;
	}
	public void setIdTipoInstrumento(int idTipoInstrumento) {
		this.idTipoInstrumento = idTipoInstrumento;
	}
	public boolean isEquipoTipoInstrumento() {
		return equipoTipoInstrumento;
	}
	public void setEquipoTipoInstrumento(boolean equipoTipoInstrumento) {
		this.equipoTipoInstrumento = equipoTipoInstrumento;
	}
}
```

Clase InstrumentoAction.java

c) **Vista** – En la vista se trata todo lo que tiene que ver con interacción con el usuario. En nuestro caso son los archivos *jsp* que usamos para desplegar los resultados de consultas. En los catálogos de la solución de software las vistas se dividen en tres partes que a continuación se explican:

1) Pantalla principal que se usa para cargar el *decorator* con encabezado, pie de página, etiquetas (*tags*), etc. En esta pantalla siempre se incluyen las librerías de *jquery* y regularmente traen un filtro de búsqueda en caso de que aplique.
2) Resultado de la consulta, es decir, se dibuja la tabla que muestra el resultado de la búsqueda, en el caso de instrumentos mostraría el listado de los mismos (con o sin filtro). Este jsp siempre tiene que estar en una carpeta que se excluya del *decorator*.

3) Pantalla de edición/ingreso de datos, es decir, una pantalla que nos permita ingresar un nuevo registro, o que nos permita modificar los datos de un registro ya existente. A esta pantalla se le aplican las definiciones del *decorator*.

Las pruebas de calidad para los mastógrafos solo se utilizaban el primer elemento de la vista.

Siguiendo las reglas aquí explicadas y según el plan de trabajo previamente asignado en el análisis, nos dimos a la tarea de ir desarrollando la solución de software.

3.3 PRUEBAS CON DATOS MUESTRA

Durante el proceso de desarrollo y conclusión de cada pantalla, se realizaban **pruebas unitarias** para comprobar que no hubiera errores de programación y que las pantallas funcionaran de forma correcta. Al detectar algún error, entonces se procedía a identificar la causa y se solucionaba el error. Una vez que las pruebas se llevaran a cabo satisfactoriamente en su totalidad, podemos decir que se ha dada por terminada la pantalla. Una vez que se concluía un módulo, entonces se realizaban **pruebas de conjunto** para comprobar que el módulo funcionara adecuadamente. Estas pruebas se realizaron con datos simulados. Si el resultado era satisfactorio se continuaba con el desarrollo de otro requerimiento. Por tratarse de un proceso en un sistema nuevo no se contó con datos base de prueba, por lo que en las pruebas de conjunto se requirió la participación al 100% de los usuarios, ya que estos conocían los datos reales de los equipos de mastografía.

Al concluir un módulo, se presentaba a los usuarios y estos podían ingresar datos reales, en un par de ocasiones se presentaron errores en la captura asociados al tipo de dato utilizado, por ejemplo por la redefinición del protocolo o por falta de comunicación. Se puede decir que las **pruebas de conjunto** las realizamos con los usuarios.

Los datos base iniciales, que los usuarios nos proporcionaron consistían en una hoja de Excel con 30 registros. Los cuales solo implicaban los instrumentos que se usaban en la UNEME de Querétaro. De haber contado con datos reales, las pruebas de conjunto se hubieran llevado a cabo antes de presentarse el módulo a los usuarios y así se hubieran previsto algunos errores. Los errores detectados fueron solucionados al igual que aquellos que se presentaron en las **pruebas unitarias**.

3.4 SOLUCIÓN A PROBLEMAS DETECTADOS

La solución de problemas se iba dando según éstos se fueran presentando, es decir, solo se reaccionó al error, ya fuera de sistema o cambio de requerimiento. Los dos eran tratados de la misma manera, lo cual fue impactando en nuestros tiempos de desarrollo haciendo que estas correcciones se hicieran durante el desarrollo de otros módulos y se acabaron resolviendo en paralelo. Fue tan significativo el impactó que se necesitaron dos meses adicionales en el desarrollo de la solución de software.

Al final del desarrollo se clasificaron los problemas en tres categorías, esto dio el tiempo necesario para la atención de errores lo que permitió darle buen término al desarrollo de la solución de software:

Categoría	Tratamiento
a) Errores del sistema	Se revisa causa del error, se estima el tiempo para su resolución, se proceden con las adecuaciones y se agenda una cita para que el usuario revise las correcciones.
b) Nuevo requerimientos c) Modificación de requerimiento inicial	Se le comenta al usuario que por tratarse de algo nuevo y no por no ser parte del alcance inicial se resolverá en una etapa posterior.

4. IMPLEMENTACIÓN

La implementación pertenece a la fase de Despliegue de la metodología implementada. En esta parte se realizan:

- Pruebas integrales.
- Manual de usuario.
- Capacitación.
- Liberación del software al usuario.

4.1 FASE DE TRANSICIÓN

De acuerdo a la metodología que se usó, la fase de transición entre desarrollo y la implementación fue de forma evolutiva, es decir, cada vez que se concluía un módulo o una prueba de calidad, esta se le mostraba a los usuarios, se realizaban pruebas con los mismos y si el funcionamiento era adecuado se liberaba el módulo.

Aunque ya se habían detectado y corregido los errores durante la construcción del sistema, nuestros usuarios seguían realizando cambios a sus protocolos del **Manual de Control de Calidad**, lo cual implicó cambios a los requerimientos iníciales.

Debido a la restricción de tiempo que se tuvo y dado que ya llevábamos 2 meses adicionales de trabajo, se optó por finiquitar la versión 1.0 del sistema, dejando para una siguiente versión, tanto los nuevos requerimientos como las modificaciones a los requerimientos iníciales que los usuarios continuaban haciendo en este nivel del proceso.

4.2 PUESTA A PUNTO

En esta parte de la implementación, nos dimos a la tarea de indicarle al **CNEGySR** las características que debía tener el ambiente de producción.

El sistema fue desarrollado en *Java SDK* versión 6 o superior, con *Struts* versión 2.3.3. El servidor de aplicaciones que se utilizó fue *Tomcat* versión 6.0.32 y la base de datos que utilizamos fue *MySQL* versión 5.

Una de las ventajas de utilizar estas características de software es que la plataforma para instalar el sistema puede ser en *Windows* o basados *en Unix (HP-UX, Linux, OSX*, etc.). Incluso en el CNEGYSR se instaló en un servidor con Ubuntu 12.

Para instalar el sistema se siguieron los siguientes pasos:

1. Instalar el JDK en el servidor.
2. Configurar el JAVA_HOME.
3. Instalar la base de datos MySQL.
4. Restaurar el respaldo de los datos iniciales del sistema (*carga inicial*).
5. Copiar o instalar los archivos de Tomcat (*Servidor de aplicaciones*).
6. Copiar el sistema en el directorio web-inf del Tomcat.
7. Editar el archivo ..\web-inf\classes\hibernate.cfg.xml y cambiar los datos de conexión a la base de datos.
8. Iniciar el servicio del servidor de aplicaciones (Tomcat)

La ruta para ingresar al sistema sería:

http://ipdelServidor:8080/ccMasto/login

Todo este procedimiento es explicado en mayor detalle en el **Manual de Instalación** que se le entrego al usuario.

Si se siguen los pasos anteriores, se tendrán configurados los datos de usuario del **Administrador del Sistema**. Con este usuario es posible empezar a dar de alta usuarios según las necesidades del **CNEGySR**.

4.3 MANUAL DE USUARIO

En nuestra experiencia en el área de sistemas, y con base en sistemas anteriores, hemos visto que los manuales de usuarios tradicionales son poco prácticos para los usuarios, ya que los manuales escritos rara vez son consultados por los usuarios finales. Por tal razón se propuso realizar video-manuales, los cuales pueden llegar a ser más prácticos. Estos video-manuales, consisten en videos descriptivos de la solución de software.

Para la narración de los video-manuales se usó el software *Loquendo TTS Director 7* y básicamente para este software se prepararon guiones, posteriormente este software reproduce con voz propia lo que se escribió.

A continuación una vez que se tiene lista la narración, nos preparamos a realizar el video en sí. El video se grabó con el software de *Camtasia Studio 7*. Este software se usa para edición de video, así como para tomar el video y/o audio de lo que estemos haciendo en la computadora (*Screen Capture*). Iniciamos la narración de la explicación de la pantalla y en sincronía con la misma se ejecutaron las acciones. Al terminar la narrativa, se detuvo la grabación del video y se verificaba si el video había sido grabado con los niveles de audio adecuado.

Una de las complicaciones de realizar los manuales de esta forma, es que hay que coincidir adecuadamente el video con lo que la narración está diciendo, y si en algún momento nos equivocamos se vuelve a comenzar.

4.4 CAPACITACIÓN

Para la capacitación se diseñó una matriz para ver que pantallas se iban a mostrar por perfil y con esto plantear un calendario de capacitación. La matriz quedó de la siguiente forma:

		Perfiles			
Módulo	**Requerimientos**	**Administrador Sistema**	**Físico Médico**	**Usuario**	**Administrativo**
Catálogos	Pantalla de mantenimiento de entidad federativa				
	Pantalla de mantenimiento de Unidad Médica				
	Pantalla de mantenimiento de Prueba				
	Pantalla de mantenimiento de medida de tiempo				
	Pantalla de mantenimiento de frecuencia				
	Pantalla de mantenimiento de tipo de instrumento				
	Pantalla de mantenimiento de Instrumento				
	Pantalla de mantenimiento Posición control de densidad CAE				
	Pantalla de mantenimiento Modo CAE				
	Pantalla de mantenimiento Modo de exposición				
	Pantalla de mantenimiento de tipo				

	de mastógrafo				
	Pantalla de mantenimiento de ánodo/filtro				
	Asociación de tipos mastógrafo con ánodo/filtro				
	Pantalla de mantenimiento de marca				
	Pantalla de mantenimiento de modelo				
	Asociación de tipos de instrumento con marcas				
	Pantalla de mantenimiento de tipo de programación				
	Pantalla de mantenimiento Posición CAE				
	Pantalla de mantenimiento Posición del detector del CAE				
	Pantalla de mantenimiento Posición del selector de densidad del CAE				
Gestión de pruebas	Pantalla de definición de pruebas				
	Pantalla de asociación de definición de pruebas con unidades médicas				
	Pantalla de asociación de instrumentos con definición de prueba				
	Proceso de programación de la prueba				
	Reporte de Resumen de control				

	Actualizar estado de las pruebas				
	Monitorear pruebas				
	Pantalla de generación de pruebas				
	Proceso de generación de pruebas				
	Pantalla de calendario de monitoreo de pruebas				
Capturar pruebas	Cabecera de pruebas				
	Pantalla de Evaluación de Fuerza de compresión				
	Pantalla de Análisis de Estudios Repetidos				
	Pantalla de Revisión visual del Equipamiento e Instalaciones				
	Pantalla de Constancia en la Homogeneidad del Receptor				
	Pantalla de Constancia de Funcionamiento Global del CAE				
	Pantalla de Artefactos en mastógrafos DR y CR				
	Pantalla de Evaluación de Monitores				
	Pantalla de Evaluación de Impresoras				
	Pantalla de Constancia de la Calidad de la Imagen				
	Pantalla de Procesador de Película				
	Pantalla de Calidad de Imagen				
	Pantalla de Cuarto Oscuro				

	Pantalla de Contacto película-pantalla				
	Pantalla de Análisis de Retención del Fijador en la Película				
	Impresión de Evaluación de Fuerza de compresión				
	Impresión de Análisis de Estudios Repetidos				
	Impresión de Revisión visual del Equipamiento e Instalaciones				
	Impresión de Análisis de Retención del Fijador en la Película				
	Impresión de Constancia de Funcionamiento Global del CAE				
	Impresión de Constancia en la Homogeneidad del Receptor				
	Impresión de Artefactos en mastógrafos DR y CR				
	Impresión de Constancia de la Calidad de la Imagen				
	Impresión de Evaluación de Monitores				
	Impresión de Evaluación de Impresoras				
	Graficar pruebas de Procesador de Película				
	Graficar pruebas de Calidad de Imagen				
	Graficar pruebas de Constancia de Funcionamiento Global del CAE				

	Graficar pruebas de Constancia de la Calidad de la Imagen				
	Graficar pruebas de Evaluación de Impresoras				
Control de Acceso	Pantalla de control de acceso				
	Proceso de verificación de usuario				
Administración de usuarios	Pantalla de mantenimiento de usuarios				
	Asociar perfil al usuario				
	Asociar unidad al usuario				
	Pantalla de mantenimiento de perfiles				

Basado en la matriz anterior, se calendarizó el programa de capacitación que se le entrego al usuario. A continuación se presenta el calendario de capacitación según la matriz:

Día	**Requerimiento**	**Administrador**	**Físico Médico**	**Usuario**	**Administra-tivo**	**Tiempo (horas)**
Día 1	Introducción a la capacitación					0,5
	Pantalla de control de acceso					0,5
	Proceso de programación de la prueba					2
	Reporte de Resumen de control					1
	Pantalla de calendario de monitoreo de pruebas					1
					Total	**5**

Día	Requerimiento	Administrador	Físico Médico	Usuario	Administrativo	Tiempo (horas)
Día 2	Proceso de verificación de usuario					**0,5**
	Pantalla de mantenimiento de usuarios					**1**
	Asociar perfil al usuario					**1,5**
	Asociar unidad al usuario					**1,5**
	Pantalla de mantenimiento de perfiles					**1**
	Pantalla de mantenimiento de entidad federativa					**0,5**
	Pantalla de mantenimiento de Unidad Médica					**0,5**
	Pantalla de mantenimiento de medida de tiempo					**0,5**
	Pantalla de mantenimiento de frecuencia					**0,5**
					Total	**7,5**

Día	Requerimiento	Administrador	Físico Médico	Usuario	Administrativo	Tiempo (horas)
Día 3	Pantalla de mantenimiento de marca					0,5
	Pantalla de mantenimiento de modelo					0,5
	Asociación de tipos de instrumento con marcas					1
	Monitorear pruebas					1
					Total	**3**

Día	Requerimiento	Administrador	Físico Médico	Usuario	Administrativo	Tiempo (horas)
Día 4	Pantalla de mantenimiento de Prueba					0,5
	Pantalla de mantenimiento de tipo de instrumento					0,5
	Pantalla de mantenimiento de Instrumento					1,5
	Pantalla de mantenimiento Posición control de densidad CAE					0,5
	Pantalla de mantenimiento Modo CAE					0,5
	Pantalla de mantenimiento Modo de exposición					0,5
	Pantalla de mantenimiento de tipo de mastógrafo					0,5
	Pantalla de mantenimiento de ánodo/filtro					0,5
	Asociación de tipos mastógrafo con ánodo/filtro					0,5
	Pantalla de mantenimiento de tipo de programación					0,5
	Pantalla de mantenimiento Posición CAE					0,5
	Pantalla de mantenimiento Posición del detector del CAE					0,5
					Total	**7**

Día	Requerimiento	Administrador	Físico Médico	Usuario	Administrativo	Tiempo (horas)
Día 5	Pantalla de mantenimiento Posición del selector de densidad del CAE					0,5
	Pantalla de definición de pruebas					1,5

	Pantalla de asociación de definición de pruebas con unidades médicas					1
	Pantalla de asociación de instrumentos con definición de prueba					1
	Actualizar estado de las pruebas					1
	Pantalla de generación de pruebas					1
	Proceso de generación de pruebas					1
					Total	**7**

Día	**Requerimiento**	**Administrador**	**Físico Médico**	**Usuario**	**Administrativo**	**Tiempo (horas)**
Día 6	Cabecera de pruebas					1
	Pantalla de Evaluación de Fuerza de compresión					1
	Pantalla de Análisis de Estudios Repetidos					1
	Pantalla de Revisión visual del Equipamiento e Instalaciones					1
	Pantalla de Constancia en la Homogeneidad del Receptor					1
	Pantalla de Constancia de Funcionamiento Global del CAE					1
	Pantalla de Artefactos en mastógrafos DR y CR					2
					Total	**8**

Día	**Requerimiento**	**Administrador**	**Físico Médico**	**Usuario**	**Administrativo**	**Tiempo (horas)**
Día 7	Pantalla de Evaluación de Monitores					1

	Pantalla de Evaluación de Impresoras					1
	Pantalla de Constancia de la Calidad de la Imagen					1
	Pantalla de Procesador de Película					1
	Pantalla de Calidad de Imagen					1
	Pantalla de Cuarto Oscuro					1
	Pantalla de Contacto película-pantalla					1
	Pantalla de Análisis de Retención del Fijador en la Película					1
					Total	**8**

Día	**Requerimiento**	**Administrador**	**Físico Médico**	**Usuario**	**Administrativo**	**Tiempo (horas)**
Día 8	Impresión de Análisis de Estudios Repetidos					0,5
	Impresión de Revisión visual del Equipamiento e Instalaciones					0,5
	Impresión de Análisis de Retención del Fijador en la Película					0,5
	Impresión de Constancia de Funcionamiento Global del CAE					0,5
	Impresión de Constancia en la Homogeneidad del Receptor					0,5
	Impresión de Artefactos en mastógrafos DR y CR					0,5
	Impresión de Constancia de la Calidad de la Imagen					0,5
	Impresión de Evaluación de Monitores					0,5

	Impresión de Evaluación de Impresoras					0,5
	Graficar pruebas de Procesador de Película					0,5
	Graficar pruebas de Calidad de Imagen					0,5
	Graficar pruebas de Constancia de Funcionamiento Global del CAE					0,5
	Graficar pruebas de Constancia de la Calidad de la Imagen					0,5
	Graficar pruebas de Evaluación de Impresoras					0,5
					Total	**7**

Durante la capacitación nos encontramos con los problemas típicos de esta actividad, es decir, los usuarios llegaban tarde, por lo que la capacitación se atrasaba, algunos usuarios se salían de la sala de capacitación antes de que ésta acabara, en otras ocasiones estos eran requeridos para atender otros asuntos y otras tantas faltaban a dichas capacitaciones. Lo que esto provoca es que los usuarios no se capaciten adecuadamente, y que sus dudas se incrementen en el momento de la operación de la solución de software.

Otro de los inconvenientes con los que nos enfrentamos al momento de la capacitación, es la falta de experiencia en el manejo de equipos de cómputo, tuvimos que dedicar algunos minutos para explicar conceptos básicos de cómputo que con otro tipo de usuarios no hubiera sido necesario mencionar.

4.5 MONITOREO DEL SOFTWARE

Como parte de la liberación del software, se hizo una entrega parcial en la UNEME de Querétaro. El objetivo de esta entrega era la realización y monitoreo de una prueba piloto en dicha clínica. La entrega parcial consistió en liberar el sistema funcionando, pero solo para los protocolos de calidad diseñados para mastógrafos digitales, ya que esa clínica contaba únicamente con mastógrafos de ese tipo.

En la presentación de la solución de software que se impartió al personal técnico radiólogo y al personal de apoyo de la UNEME, surgieron comentarios con respecto al mismo y así como dudas referentes al **Manual de Control de Calidad**. En lo concerniente a la solución de software tomamos notas de los comentarios y las observaciones que nos parecieron pertinentes, estas se platicaron con el usuario generando cambios en la solución, los cuales se integraron en la versión final.

Una actividad importante antes de entregar formalmente la solución de software, fue reunirnos con los físicos médicos del **CNEGySR** y con los usuarios de la clínica para dar de alta todos los instrumentos que se utilizan en la UNEME. Una vez terminada la carga de esta información fue posible comenzar a realizar la captura de los protocolos de calidad a través de la solución de software.

De esta experiencia surgieron mejoras para ser integradas a la solución de software para complementar la operación. En la UNEME cuando llegaba a quedarse sin internet, el registro de los protocolos de calidad podía detenerse debido a que la solución de software esté centralizada en los servidores del **CNEGySR**. Se descartó la posibilidad de descentralizar la solución de software, esto porque los administrativos del **CNEGySR** requieren evaluar la información de todos los equipos de mastografía a nivel nacional de manera oportuna, y de forma centralizada esto se vuelve más factible.

4.6 SOPORTE

Con los usuarios se llegó al acuerdo que durante 6 meses se brindará soporte, solo se solicitó que fuera de forma más organizada, las solicitudes de soporte serán enviadas vía correo electrónico a un usuario administrativo que a su vez, reenviará la solicitud agregando el nivel de prioridad. Los niveles de prioridad se establecieron del 1 al 5, siendo el 5 el más importante y el uno el más bajo.

Con base en la prioridad se ordenan y revisan las solicitudes, para estimar el tiempo de respuesta. Además se coloca una nueva categoría del 1 al 4, donde uno implica un cambio significativo, dos es un cambio normal, tres implica un cambio de vista y cuatro solo requiere una explicación al usuario. Se acordó que la categoría uno implica tiempo en el desarrollo y se tendría que evaluar sí el cambio se incluye en una versión posterior.

Estas solicitudes se denominan “tickets” con un número consecutivo, ya que esto permite tener un control sobre las solicitudes.

Para la atención de los tickets, se clasifican en pendiente y atendido. El estatus de pendiente es cuando solo se ha evaluado la solicitud pero no se ha resuelto. El estatus de atendido se aplicado cuando la solicitud fue resuelta.

Como parte del seguimiento a las solicitudes, se pactó con el usuario tener una junta mensual donde se entregara una lista de las solicitudes, y un porcentaje de atención diario. Se revisan los tiempos de entrega de las solicitudes pendientes. Y si alguna solicitud rebasa la fecha estimada de solución, se replantea con el usuario para establecer una nueva fecha.

Conclusiones

- Se construyó satisfactoriamente el SSCEM de acuerdo a la solución propuesta.

- El SCCEM le permite al CNEGYSR tener un control preciso y a tiempo de la calidad de los equipos de mastografía. Con este sistema y con base en los procesos de calidad generados por los físicos médicos, es posible tener por primera vez los indicadores de calidad de los equipos de mastografía en tiempo real. Este tipo de control se tiene en algunos países como Estados Unidos y Brasil, está es la primera vez que se tienen en México, lo cual es un gran beneficio para seguir ofreciendo altos niveles de calidad al pueblo de México en el servicio de mastografía.

- Consideramos también que existen diferentes tecnologías las cuales pueden darnos una solución, siendo nuestra responsabilidad como ingenieros el decidir cuál o cuáles herramientas se ajustan mejor a las condiciones con las que se cuenta dentro de la organización, además se toma en cuenta los costos y los recursos humanos.

- Otro punto importante es la planeación e integración del proyecto, así como el equipo de trabajo para cumplir los objetivos del mismo con los mejores rendimientos.

- Al haber desarrollado este sistema con el framework de Struts, nos permitió conocer y aprender una herramienta que laboralmente nos pueda dar más oportunidades ya que Struts es uno de los frameworks de Java más utilizados.

- También la realización de este proyecto nos permitió conocer las herramientas ORM, las cuales presentan grandes ventajas a la hora de desarrollar sistemas. Anteriormente habíamos trabajado en sistemas en donde no se mapeaba la base de datos, lo cual tenía serias desventajas, una de ellas era que al cambiar una tabla era necesario buscar en todos los lugares donde se encontraban las consultas nativas para realizar los cambios, además de que el sistema en cuestión quedaba fijo para esa base de datos. Aprendimos que los ORM le dan independencia al sistema con respecto a la base de datos.

- También con este sistema conocimos las ventajas de seguir el modelo MVC, ya que así tenemos separados los 3 componentes de un sistema, lo cual permite que el mantenimiento del mismo sea más ordenado, y al mismo tiempo si se tratara de un proyecto grande, el MVC nos permitiría tener equipos de trabajo desarrollando cada uno de los componentes casi de forma independiente.

- Por nuestra parte queda asentado que para el desarrollo de software se puede contar con diferentes metodologías, las cuales se pueden adaptar de acuerdo a las necesidades del equipo de trabajo o a los objetivos del proyecto.

Glosario

Físico Médico: se llama al personal con la formación en la Maestría de Física Médica que incluye aplicaciones de la física a la medicina en la prevención, diagnóstico y tratamiento de las enfermedades humanas, así como en la investigación médica para la promoción y conservación de la salud. (Médica, 2013)

Técnico Radiólogo: persona encargada de realizar la toma de las radiografías en este caso de las mastografías.

Normas Oficiales Mexicanas (NOM): son un conjunto de reglas emitidas por el gobierno para prevenir e identificar riesgos, además de evaluarlos. El proceso de elaboración de las NOM es por medio de Comités Técnicos, los cuales están integrados por representantes de todos los sectores interesados, como las dependencias gubernamentales correspondientes y expertos externos provenientes de la academia, de las cámaras industriales o de colegios de profesionistas. (Profeco, 2013)

Mastografía o mamografía: es una radiografía que puede mostrar la presencia de formaciones o tumores antes de tenga un tamaño suficiente para que la mujer o el médico puedan percibir una malformación mediante la exploración de la glándula mamaria. (PortalesMedicos, 2011)

Radiografía: es la impresión en una placa fotográfica con una mínima cantidad de radiación, que se hace pasar por una zona del cuerpo. Se dejan pasar diferentes cantidades de radiación dependiendo del tipo de tejido del organismo. Cuando se mira una radiografía y conociendo la imagen que se debe tener en una radiografía normal, pueden identificar imágenes que ayuden al diagnóstico (quistes, tumores, aumento o disminución de tamaño de órganos, etc). (Salud I. N., 2012)

Ionización: hace referencia a la disociación de moléculas en diferentes iones o a la transformación de una molécula o de un átomo en un ion.

Ultrasonografía: es técnica de imagen basada en la diferente capacidad de los tejidos para reflejar o refractar las ondas de ultrasonido emitidas por el equipo. Estas son emitidas y detectadas por un equipo que, mediante la codificación, en un plano, de los diferentes puntos de reflexión generados por el tejido, los representa en una imagen en gama de grises, de forma proporcional a la intensidad de la reflexión, según su frecuencia y el tiempo en que son detectados. (Médicas, 2011)

Cáncer: es una enfermedad provocada por un grupo de células que se multiplican sin control y de manera autónoma, invadiendo localmente y a distancia otros tejidos.

Tumor: es una masa de tejido de una parte del organismo cuyas células sufren un crecimiento anormal. Los tumores puede ser cancerosos benignos o malignos. (usted, 2012)

Biopsia: es la extracción de una muestra de tejido de un organismo vivo para su estudio y análisis microscópico posterior, con la ayuda del instrumento adecuado dependiendo de cada caso.

Tamizaje (cribado): es una estrategia que se aplica sobre una población para detectar una enfermedad en individuos sin síntomas. El fin del tamizaje es identificar enfermedades de manera temprana dentro de una comunidad para reducir los efectos provocados por una enfermedad uno como el costo. Para que se considere como una medida preventiva debe cumplir con los criterios de Frame y Carlson que establecen:

- La causa común de morbimortalidad.
- Detectable y tratable en etapa presintomática.
- Los tests para diagnosticarla deben ser efectivos y eficaces.
- El tratamiento temprano debe ser mejor que el tratamiento en la etapa sintomática o de diagnóstico habitual.
- El daño potencial de la intervención debe ser menor que el del tratamiento no precoz. (Wikipedia, 2013)

Prevención: acción o disposición que se toma de manera anticipada para evitar una situación o hecho.

Diagnóstico: es la identificación de una situación, hecho o enfermedad mediante el examen de los síntoma que presenta el paciente para buscar una solución.

Calidad: conjunto de propiedades esenciales de algo, que permite calificar su valor, buscando la excelencia. En nuestro caso buscamos la calidad en el funcionamiento de los equipos de mastografía para mejorar la imagen de las mastografías.

Eficiencia: es la habilidad para realizar adecuadamente una función ó tarea.

Eficacia: es la destreza para obtener el resultado de un objetivo deseado.

Pruebas: son criterios o reglas definidas para evaluar los equipos de mastografía así como el material.

Protocoló: indica la forma o procedimiento como se deben de aplicar las pruebas.

Indicadores de pruebas: son los criterios de aceptación definidos en cada prueba para calificar cada elemento a evaluar.

Nivel óptimo de un equipo: de acuerdo a los criterios definidos en las pruebas o protocolos ya que cada una de acuerdo al criterio definido nos indicara si está funcionando bien o mal el equipo. (CNEGyRS, 2012)

Sistema Integral de Calidad en Salud (SICALIDAD): estrategia del Gobierno Federal instrumentada por la Secretaría de Salud para mejorar la calidad de los servicios de salud. (CNEGyRS, 2012)

Referencias

Agency, I. A. (2013). *Protección Radiologica de los Paciente.* Recuperado el 19 de Septiembre de 2013, de International Atomic Energy Agency: https://rpop.iaea.org/RPOP/RPoP/Content-es/InformationFor/HealthProfessionals/1_Radiology/Radiography.htm

ALegsa. (1998). *Definición de Framework.* Recuperado el Febrero de 2013, de Alegsa.com: http://www.alegsa.com.ar/Dic/framework.php

B., B. (1998). *A Sipiral Model for Software Devlopment and Enhancement* (Vol. 12).

Booch, G., Ivar, J., & James, R. (2009). *El lenguaje Unificado de Modelado* (Segunda ed.). España: Pearson.

California, U. A. (Junio de 2004). *El Proceso Unificado de Desarrollo de Software (RUP).* Recuperado el Febrero de 2013, de Servidor yaqui: http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm

Castellano, P. e. (2013). *Manual Básico de Struts.* Recuperado el Marzo de 2013, de Programación en Castellano: http://www.programacion.com/articulo/manual_basico_de_struts_156

Castellano, P. e. (2012). *Persistencia de Objetos Java: El Camino hacia Hibernate.* Recuperado el 2013, de Programacion en castellano: http://www.programacion.net/articulo/persistencia_de_objetos_java:_el_camino_hacia_hibernate_251

CDTN. (2008). *Controle da Qualidade dos Exames de Mamografia Realizados em Instituições, Clínicas, Hospitais e Laboratórios de Belo Horizonte e Minas Gerais.* Recuperado el Marzo de 2013, de CDTN 2008 - Sistema Atalanta : http://mamografia.cdtn.br/gestao-relatorios-externos/consultar-unidade/unidade/305

CDTN. (s.f.). *Controle da Qualidade dos Exames de Mamografia Realizados em Instituições, Clínicas, Hospitais e Laboratórios de Belo Horizonte e Minas Gerais.* Obtenido de Controle da Qualidade dos Exames de Mamografia Realizados em Instituições, Clínicas, Hospitais e Laboratórios de Belo Horizonte e Minas Gerais: http://mamografiamg.cdtn.br/gestao-relatorios-externos/aprovacao-mensal

CDTN. (2009). *Controle da Qualidade em Mamografia .* Recuperado el Noviembre de 2012, de Centro de Desenvolvimento da Tecnologia Nuclear : http://www.cdtn.br/linhas_de_atuacao/Projetos/controle-da-qualidade-em-mamografia

Ceballos, F. J. (2000). *Java2 curso de programación.* México: AlfaOmega Grupo Editor.

CNEGyRS. (2012). *Manual de Control de Calidad en Mastografía* (Primera ed.). México D.F.: Secretaria de Salud.

Confluence, A. (21 de Agosto de 2008). *Apache Struts 2 Documentation OGNL.* Recuperado el Marzo de 2013, de Apache Struts 2 Documentation: http://struts.apache.org/2.0.11.1/docs/ognl.html

Date, C. J. (2001). *Introducción a los sistemas de bases de datos.* México: Pearson.

decodigo.com. (2012). *Struts1 vs Struts2.* Recuperado el Abril de 2013, de decodigo.com: http://www.decodigo.com/2009/07/struts1-vs-struts2.html

Federación, D. O. (8 de Noviembre de 2011). *Norma Oficial Mexicana NOM-011-SSA2-2011.* Recuperado el Noviembre de 2012, de www.dof.gob.mx: http://dof.gob.mx/nota_detalle.php?codigo=5223519&fecha=08/12/2011

Federación, D. O. (9 de Junio de 2011). *Norma Oficial Mexicana NOM-041-SSA2-2011, Para la prevención, diagnóstico, tratamiento, control y vigilancia epidemiológica del cáncer de mama.* Recuperado el Enero de 2013, de www.dof.gob.mx: http://dof.gob.mx/nota_detalle.php?codigo=5194157&fecha=09/06/2011

Fernandez, L. H. (21 de Abril de 2009). *Patrones de Diseño.* Recuperado el Febrero de 2013, de Software Guisho: http://software.guisho.com/patrones-de-diseno

Foundation, A. S. (2012). *Struts 2.* Recuperado el Mayo de 2013, de Apache Software Foundation: http://struts.apache.org/release/2.3.x/index.html

Frías, P. (16 de Agosto de 2009). *Patrón MVC - Struts versión 1 y 2.* Recuperado el Marzo de 2013, de Mundo Java: http://mundojava.blogspot.mx/2009/08/patron-mvc-struts-version-1-y-2.html

Hat, R. (2000). *Hibernate.* Recuperado el Septiembre de 2012, de Red Hat: http://www.hibernate.org/

Hernandez, J. (2007). *Patrones de Diseño.* Recuperado el Febrero de 2013, de blogger.com: http://patronesdediseno.blogspot.mx/2009/05/patron-facade.html

IARC. (s.f.). *IARC.* Recuperado el Marzo de 2013, de IARC: http://screening.iarc.fr/doc/ND7306954ENC_002.pdf

IARC. (s.f.). *International Agency of Research on Cancer.* Recuperado el Marzo de 2013, de IARC: http://www.iarc.fr/search.php?cx=009987501641899931167%3Ajwf5bx4tx78&cof=FORID%3A9&ie=UTF-8&ie=ISO-8859-1&oe=ISO-8859-1&sa=&q=system+quality+control+mamographic#gsc.tab=0&gsc.q=system%2Bquality%2Bcontrol%2Bmammography

IBM. (9 de Marzo de 2011). *Struts 2.0 with OGNL.* Recuperado el Febrero de 2013, de IBM, developerWorks: http://www.ibm.com/developerworks/library/os-struts2ognl/index.html

Ideas, D. (s.f.). *Mapeo Objeto Relacional.* Recuperado el Marzo de 2013, de Dos Ideas: http://www.dosideas.com/wiki/Mapeo_Objeto-Relacional

Identity, P. (2012). *About OGNL.* Recuperado el Marzo de 2013, de Ping Identity: http://documentation.pingidentity.com/display/PF66/About+OGNL

INCA. (s.f.). *INCA.* Recuperado el Marzo de 2013, de INCA: http://www2.inca.gov.br/wps/wcm/connect/94510f804eb684b48b339bf11fae00ee/Balan%C3%A7o+Recome

nda%C3%A7%C3%B5es+C%C3%A2ncer+de+Mama+2012.pdf?MOD=AJPERES&CACHEID=94510f804eb684b48b339bf11fae00ee

India, R. f. (19 de Enero de 2008). *Hibernate Architecture.* Recuperado el Abril de 2013, de Reserved for Rose India: http://www.roseindia.net/hibernate/hibernate_architecture.shtml

JN, W. (2013). *CRITICAL EVALUATION OF MAMMOGRAPHY IN THE MANAGEMENT OF BREAST DISEASE.* Recuperado el Febrero de 2013, de Unbound MEDLINE: http://www.unboundmedicine.com/medline/citation/14207414/MAMMOGRAPHY:_REPORT_ON_ITS_USE_IN_WOMEN_WITH_BREASTS_ABNORMAL_AND_NORMAL_ON_PHYSICAL_EXAMINATION_

Médica, M. e. (2013). *Maestría en Física Médica.* Recuperado el Marzo de 2013, de Posgrado Ciencias Físcias, UNAM: http://www.fisica.unam.mx/fismed/index.php?option=com_content&task=view&id=58&Itemid=77

Médicas, D. (2011). *Definición de Ultrasonografia.* Recuperado el 17 de Septiembre de 2013, de Definicionesdemedicina: http://www.definicionesdemedicina.com/ultrasonografia/

Mkyong. (11 de Junio de 2010). *Working with Struts 2 actions.* Recuperado el Marzo de 2013, de Mkyong.com: http://www.mkyong.com/struts2/working-with-struts-2-actions/

NCBI. (s.f.). *Articulos NCBI.* Recuperado el Mayo de 2013, de NCBI: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1409066/pdf/annsurg00932-0124.pdf

NICA. (1996-2013). *Ministério da Saúde.* Recuperado el Febrero de 2013, de NICA: http://www2.inca.gov.br/wps/wcm/connect/acoes_programas/site/home/nobrasil/programa_controle_cancer_mama/controle_garantia_qualidade_mamografia

Pacheco, J. C. (Agosto de 2012). *Qué es Inyección de dependencias.* Recuperado el Febrero de 2013, de Microsoft Developer Network: http://msdn.microsoft.com/es-es/library/jj635998.aspx

PortalesMedicos. (30 de Diciembre de 2011). *Mastografía.* Recuperado el 17 de Septiembre de 2013, de PortalesMedicos: http://www.portalesmedicos.com/diccionario_medico/index.php/Mastografia

PortalesMedicos. (30 de Diciembre de 2011). *Mastografía.* Recuperado el 17 de Septiembre de 2013, de PortalesMedicos: http://www.portalesmedicos.com/diccionario_medico/index.php/Mastografia

Pressman, R. S. (2010). *Ingeniería del Software un enfoque práctico* (séptima ed.). México: McGrawHill.

Profeco. (17 de Mayo de 2013). *La importancia de las Normas Oficiales Mexicanas.* Recuperado el 25 de Septiembre de 2013, de Portal del Consumidor Profeco: http://www.consumidor.gob.mx/wordpress/?p=5596

Rafael, J. (22 de Julio de 2008). *Modelo Vista Controlador.* Recuperado el Febrero de 2013, de neleste: http://www.neleste.com/modelo-vista-controlador/

Salud, I. N. (14 de Agosto de 2012). *Rayos X.* Recuperado el 12 de Septiembre de 2013, de Medline Plus Información de salud para usted: http://www.nlm.nih.gov/medlineplus/spanish/ency/article/003337.htm

Salud, S. d. (2002). *Documentos.* Recuperado el Septiembre de 2012, de www.salud.gob.mx: http://www.salud.gob.mx/unidades/cdi/documentos/DOCSAL7216.pdf

Salud, S. d. (2007-2012). *Programa de Acción Especifíco 2007-2012.* Recuperado el Noviembre de 2012, de calidad.salud.gob.mx: http://www.calidad.salud.gob.mx/doctos/calidad/pa_sicalidad.pdf

Schmuller, J. *Aprendiendo UML en 24 horas.* Prentice Hall.

Sommerville, I. *Ingenieria del Software.* Prentice-Hall.

Ttutorialspoint. (2012). *Hibernate Criteria Queries.* Recuperado el Febrero de 2013, de Tutorialspoint: http://www.tutorialspoint.com/hibernate/hibernate_criteria_queries.htm

usted, M. P. (3 de Septiembre de 2012). *Tumor.* Recuperado el 17 de Septiembre de 2013, de Institutos Nacionales de la Salud: http://www.nlm.nih.gov/medlineplus/spanish/ency/article/001310.htm

usted, M. P. (3 de Septiembre de 2012). *Tumor.* Recuperado el 17 de Septiembre de 2013, de Institutos Nacionales de la Salud: http://www.nlm.nih.gov/medlineplus/spanish/ency/article/001310.htm

Viana, I. (6 de Agosto de 2008). *Object Freezer-Relational. Mapeo objeto-relacional.* Recuperado el Marzo de 2013, de Fundación del Software Libre: http://objectfreezer-r.sourceforge.net/2%20Mapeo%20objeto-relacional.html

W3Schools. (1998-2013). *W3Schools Introduction to XML.* Recuperado el Febrero de 2013, de W3Schools: http://www.w3schools.com/xml/xml_whatis.asp

Wikipedia. (18 de julio de 2013). *Screening (medicina).* Recuperado el 17 de Septiembre de 2013, de Wikipedia: http://es.wikipedia.org/wiki/Screening_%28medicina%29