Práctica Empresarial en la Fundación Cardiovascular de Colombia Basada en la Unificación de Aplicativos en un portal WEB para la Sección de TELEMEDICINA. Tele-Uci, Tele-Consulta, Tele –Apoyo Diagnóstico, Monitoreo, Chat, Video Paciente

Julián Guillermo Hernández Gómez

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2009

Práctica Empresarial en la Fundación Cardiovascular de Colombia Basada en la Unificación de Aplicativos en un portal WEB para la Sección de TELEMEDICINA. Tele-Uci, Tele-Consulta, Tele –Apoyo Diagnóstico, Monitoreo, Chat, Video Paciente

Julián Guillermo Hernández Gómez

Trabajo de grado realizado como cumplimiento de los requisitos para Optar por el título de Ingeniero de Sistemas.

Prof. Elberto Carrillo Rincón
Director Práctica Empresarial

Ing. Cesar Alberto Bayona Ríos
Tutor Práctica Empresarial
Administrador WEB

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2009

“Lo Urgente Jamás Debe Desplazar lo Prioritario...”

Dedicado a Dios que siempre ha sido mi compañero incondicional y mi esperanza en todo momento de este camino llamado vida, gracias a ti Señor por permitirme disfrutar de este éxito que con tanto esfuerzo he conseguido.

A toda mi familia, en especial a mi madre Minerva Gómez de Hernández; a quien le debo todo lo que soy, por ese gran apoyo que me ha brindado durante todos estos años.

A mi padre Guillermo Hernández Díaz; por su ejemplo de trabajo y disciplina que forjó en mí.

A mis hermanos Derly Johanna y Andrés Felipe; los quiero mucho, y gracias por todos los momentos tan maravillosos que juntos compartimos, siempre los llevaré en mi corazón.

Y a mi novia Rocío Bayona; que apareció en el momento más importante de mi vida, gracias por brindarme ese gran amor, que es el motor de mi vida, por su comprensión, perseverancia y paciencia conmigo. Cada día doy gracias a la vida por permitirme conocer y querer a una mujer tan maravillosa.

AGRADECIMIENTOS

Agradezco al profesor Elberto Carillo Rincón, director del proyecto, por apoyarme, por confiar en mí, por siempre estar en disposición de escuchar, y por compartir sus sabias ideas en el desarrollo del proyecto.

A la Fundación Cardiovascular de Colombia por darme la oportunidad de recibir mi experiencia laboral, por brindarme la confianza para realizar este trabajo.

A los Ingenieros Jaider Rodríguez y Silvia Vargas, que estuvieron pendientes de mi aprendizaje y capacitación en todo momento, que junto ellos aprendí un sinfín de cosas que tendré en cuenta no solo en mi vida laboral, sino también en mi vida personal.

Al ingeniero Cesar Bayona, quien antes de ser un compañero laboral y un tutor, es un gran amigo; quien es un gran ejemplo de compromiso y lealtad, a seguir.

A Slendy Bohórquez, quien fue la amiga que se convirtió en el contacto incondicional y puente entre fundación y universidad.

A la Familia Bayona Ríos, mi segundo hogar, que me acogió como un hijo más.

A mis amigos que me acompañaron durante diferentes etapas de mi vida, y me brindaron su apoyo incondicional y la energía para seguir adelante; ésto va dedicado a ustedes.

Y a todas esas personas, familiares, amigos, y compañeros que de alguna u otra forma intervinieron para lograr esta meta; a todos ellos un infinito agradecimiento.

CONTENIDO

LISTA DE FIGURAS

RESUMEN

TITULO: Práctica Empresarial en la Fundación Cardiovascular de Colombia Basada en la Unificación de Aplicativos en un portal WEB para la Sección de TELEMEDICINA. Tele-Uci, Tele-Consulta, Tele –Apoyo Diagnóstico, Monitoreo, Chat, Video Paciente.

AUTOR: JULIÁN GUILLERMO HERNÁNDEZ GÓMEZ

PALABAS CLAVES:
FCV, SAHIWEB, Modelo Vista Controlador, Framework Struts, SQL 2005, MyEclipse.

DESCRIPCIÓN:

La Fundación Cardiovascular de Colombia, desde su creación en 1986, ha trabajado con criterios de excelencia, innovación y un alto sentido social, se ha concentrado en brindar a los usuarios la mejor atención y una garantía de calidad en la presentación de los servicios de salud.

El desarrollo de la práctica giró en torno a tres pilares fundamentales: en primer lugar, apoyar el acoplamiento de los procesos y procedimientos actuales, con las nuevas metodologías de desarrollo software y herramientas tecnológicas enfocadas al ambiente Web; en segundo lugar, utilizar los conocimientos adquiridos en ingeniería de software para aportar en la construcción de las aplicaciones para el nuevo proyecto que consiste en un portal Web, para facilitar la comunicación entre Doctores de diferentes lugares del país; por último se realizaron tareas adicionales como atención y capacitación de usuarios y Help Desk, contempladas dentro del manual de responsabilidades del cargo asignado, y que permitieron crecimiento profesional del estudiante en la práctica.

La práctica empresarial permite desarrollar en el estudiante, experiencia y habilidades en el entorno laboral que le permitirán desempeñar mejor sus funciones de ingeniería. Además, el uso de tecnologías tales como Hibernate y Struts facilita el desarrollo de aplicaciones Web de alta calidad.

---

* Trabajo de Grado

** Facultad de Ingeniería Fisico-Mecanicas. Escuela de Ingeniería de Sistemas e Informática.
Universidad Industrial de Santander.
Director. Elberto Carrillo Rincón
Tutor. Ingeniero Cesar Bayona Ríos.

ABSTRAC

TITLE: Business practices in the Cardiovascular Foundation of Colombia Unification-Based Applications in a Web portal for the Section of TELEMEDICINE. Tele-ICU, Tele-consultation, Tele-Support Diagnosis, Monitoring, Chat, Video Pt.

AUTHOR: JULIÁN GUILLERMO HERNÁNDEZ GÓMEZ

KEY WORDS:
FCV, SAHIWEB, Model View Controller, Framework Struts, SQL 2005, MyEclipse.

DESCRIPTION

Cardiovascular Foundation of Colombia since it's inception in 1986, has worked with criteria for excellence, innovation and high social sense has focused on providing users with the best care and quality assurance in the production of health services .

The development of practical spin around three pillars: first, supporting the coupling of processes and procedures, with new software development methodologies and technology tools focused on the Web environment, secondly, using the lessons learned in software engineering to provide in building applications for the new project is a Web portal to facilitate communication between doctors in different parts of the country, and finally were performed additional tasks such as attention and user training and Help Desk , referred to in the manual of responsibilities of the assignment, which allowed students professional growth in practice.

Business practice allows the student to develop, experience and skills in the workplace to help you better perform their duties in engineering. Furthermore, the use of technologies such as Hibernate and Struts facilitates the development of high-quality Web applications.

---

*Graduation Work
**Physical – Mechanical Engineering´s Faculty.
System Engineering and Computer Science School. UIS.
Manager. Carrillo Eilbertus Corner
Tutor. Engineer Bayona Cesar Rios.

INTRODUCCIÓN

La modalidad de práctica empresarial, para el estudiante, es una excelente oportunidad de poner en producción en una empresa todos los conocimientos adquiridos durante su carrera universitaria, en donde se medirá no solo sus cualidades intelectuales, sino también sus habilidades interpersonales, su capacidad de respuesta ante las diferentes situaciones difíciles que se le presente, la facilidad de integrarse y trabajar conjuntamente en equipo, relacionarse con clientes, entre otras muchas experiencias que le permitirán al estudiante adquirir un concepto claro sobre la vida empresarial.

Actualmente la velocidad con la que se crean nuevas tecnologías crece exponencialmente, así como la aparición de nuevas y mejores metodologías para el desarrollo de proyectos informáticos, además del auge de las aplicación orientadas a la Web, ha llevado a las empresas a desarrolladoras de software a modernizar sus perspectivas para permanecer en el mercado que cada vez se hace más competitivo.

La FCV no es ajena a estos cambios y ante la problemática que surge en el ambiente hospitalario de querer tener el seguimiento de los reportes de un paciente en todo momento y en cualquier lugar, sugerir, corregir e informar respecto a las observaciones de un paciente en tiempo real, eran unas de las más conocidas necesidades que los doctores siempre habían manifestado.

Por otra parte, en la mayoría de los procedimientos quirúrgicos y médicos siempre han surgido dudas tales como “¿será esta intervención la mejor para este paciente? ¿Este medicamento es el óptimo para este caso en particular?”.

Para ayudar a resolver estas inquietudes se han creado diversas soluciones, todas en busca de interconectar la perspectiva de diferentes doctores para lograr un control y seguimiento en tiempo real de todos los pacientes, independiente de su ubicación en el mundo.

El presente proyecto propone una solución Web de los inconvenientes descritos anteriormente, aprovechando del material humano, con capacidades, visión emprendedora y conocimientos frescos, que les permitan alcanzar sus objetivos,

en el ofrecimiento de prácticas empresariales a estudiantes que están finalizando sus estudios es una mina de recursos humanos de gran calidad técnica y de excelente perfil profesional, y con más razón si son los estudiantes de la universidad Industrial de Santander, reconocidos ampliamente a nivel regional y nacional.

El presente documento realiza la recopilación teórica y práctica de los eventos y actividades realizadas durante los seis meses de desarrollo de la práctica empresarial, en donde se documentan todas las experiencias adquiridas por el estudiante practicante, así como las conclusiones, y recomendaciones.

1. DESCRIPCIÓN DE LA PRÁCTICA

En el presente capítulo se describen aspectos generales de la organización en la que se desarrolló la práctica empresarial, tales como la razón social, reseña histórica y campo de acción, además se presenta una descripción de los objetivos, alcance del proyecto y a su vez el impacto y su viabilidad.

1.1. DESCRIPCIÓN DE LA EMPRESA

1.1.1. Nombre de la Empresa.

Razón Social:	Fundación Cardiovascular de Colombia
Nit:	No. 890.212.568-0
Tipo de Organización:	Entidad privada sin ánimo de lucro
Fecha de fundación:	1986
Domicilio:	Calle 155 N. 23-58 Sector E1 El Bosque
Ciudad:	Bucaramanga
Representante Legal:	Víctor Raúl Castillo Mantilla

1.1.2. Reseña Histórica.

Nuestra historia se remonta al año 1986 cuando un grupo de especialistas y personalidades de Bucaramanga se propuso crear una entidad privada sin ánimo de lucro dedicada a tratar las enfermedades del corazón, logrando en octubre de 1990 que un grupo de médicos iniciara las actividades de consulta y prueba de esfuerzo en la Fundación Tercera Edad de la Congregación Mariana, y las primeras cirugías cardiovasculares en la Clínica Bucaramanga.

En el año 1992 entró a formar parte de la Clínica Carlos Ardila Lulle, adquiriendo el cuarto piso, ampliando así todos los servicios diagnósticos e intervencionistas de cardiología y cirugía vascular periférica, utilizando salas de cirugía, unidad de cuidados intensivos y hospitalización de esta moderna clínica.

Posteriormente en octubre de 1997 se inauguró la nueva sede del Instituto del Corazón, un moderno edificio de 14 pisos con una capacidad de 123 camas de hospitalización distribuidas entre la unidad de Cuidados Intensivos Post-quirúrgica, unidad de Cuidado Intensivos unidad de Cuidados Intensivos Pediátrica, unidad de Cuidados Intermedios Adultos, tres pisos de hospitalización, 4 salas de cirugía, 2 salas de Hemodinámica y 1 de más del servicio de urgencias durante las 24 horas del día cumpliendo así con todos los requisitos y normas exigidas por el Ministerio de salud relacionadas con enfermedades cardiovasculares.

1.1.3. Quienes Somos.

Hoy la FCV a través de su instituto del corazón Floridablanca, está catalogada como una de las cinco mejores IPS del país, en un nivel de excelencia, tras obtener la primera acreditación en salud en Colombia.

Su experiencia a nivel administrativo y médico-científico se comparte en otras ciudades como Santa Marta, Ibagué y Manizales, donde la FCV actúa como administradora delegada de importantes clínicas; al tiempo que la población de otros 13 departamentos recibe atención en salud por telemedicina, en especialidades como cardiología, neurología, radiología y cuidado intensivo, entre otras.

En el campo científico, la FCV con su dirección de investigaciones es reconocida en los ámbitos nacional e internacional por la generación de conocimiento, la clasificación de grupos en Colciencias y un número considerable de publicaciones especializadas.

Como organización empresarial ha desarrollado 11 Unidades de negocios (UEN) que se encargaran de proveer servicios y productos de la más alta calidad al sector salud, en aéreas como: ropa e insumos quirúrgicos, equipos biomédicos, soluciones informáticas y de comunicación, administración hospitalaria y telemedicina, tomando como base la innovación y personalización de las soluciones.

Su responsabilidad social abarca importantes programas como “Corazón a Corazón y “Montañas Azules”, que ofrecen a pacientes de escasos recursos, sin seguridad social y procedentes de regiones apartadas, la cirugía, el tratamiento médico y satisfacción de sus necesidades básicas, mientras reciben la atención requerida.

1.1.4. Misión de la Empresa.

La Fundación Cardiovascular de Colombia, es una organización empresarial sin ánimo de lucro que provee servicios y productos de salud de alta calidad para el desarrollo del sector buscando permanentemente el bienestar de la comunidad.

FVC Soft es una unidad empresarial de negocios de la Fundación Cardiovascular de Colombia que construye y diseña herramientas software bajo estándares internacionales, buscando incrementar el conocimiento, la productividad y la competitividad empresarial mediante la transferencia tecnológica de las mismas.

FVC Centro Nacional de Telemedicina es una unidad empresarial de negocios de la Fundación Cardiovascular de Colombia, que busca contribuir a la optimización y modernización de los servicios de salud en Colombia con calidad, eficiencia y equidad para beneficio prioritario de las poblaciones excluidas y dispersas, a través de la incorporación de tecnologías de información y comunicación (TICS).

1.1.5. Visión de la Empresa.

En el año 2020 la Fundación Cardiovascular de Colombia será una organización reconocida a nivel nacional e internacional por la excelencia e innovación de sus productos y servicios orientados principales al sector salud.

1.1.6. Valores Corporativos.

- **Laboriosidad**

Realizar nuestro trabajo con total dedicación, interés y esmero, procurando siempre entregar lo mejor de nosotros mismos, para obtener resultados óptimos que generen satisfacción total en los clientes, utilizando adecuadamente los recursos proporcionados por la Institución. Haciendo las cosas bien desde el principio hasta el fin, observando con alto sentido ético todas las actuaciones e intervenciones en los productos y servicios que llegan hasta nuestros clientes, anticipándonos a las oportunidades de mejora que puedan llevarnos a trabajar cada días más y mejor.

- **Innovación y Creatividad**

Trabajar en pro del desarrollo personal e institucional, creando nuevas y mejores formas de hacer las cosas, manteniendo siempre una actitud de flexibilidad hacia el cambio que a su vez permita la búsqueda de soluciones hacia contratiempos inesperados que conlleven a seguir fortaleciendo la capacidad de aprendizaje continuo.

- **Trato Humanizado**

Generar confianza, emociones agradables y sentimientos humanos de buen trato a nuestros clientes y proveedores, para así permitir momentos de verdad y otorgar valor agregado en el servicio que les ofrecemos.

- **Lealtad**

Trabajar día a día demostrando un alto sentido de pertenencia y compromiso institucional hacia la FCV, uniendo esfuerzos para el cumplimiento de metas y objetivos, defendiendo el nombre de la institución, y actuando siempre con transparencia y sinceridad, siendo leales hacia las normas y valores de la institución.

- **Respeto**

Contribuir al mantenimiento de un ambiente de trabajo cordial y amable reconociendo y aceptando los derechos y las diferencias de las demás personas, cumpliendo de manera oportuna con las responsabilidades establecidas y brindando un trato considerado y cortés a las personas con las que día a día nos relacionamos, principalmente nuestros clientes.

- **Solidaridad**

Actuar con equidad orientando la labor hacia la comunidad ofreciendo apoyo y colaboración a las demás personas, trabajando con sentido de fraternalidad y unión que no sólo conlleve a la obtención de logros y metas personales, sino propendiendo además al cumplimiento de objetivos que promuevan el desarrollo y progreso institucional.

- **Honestidad**

Actuar con la verdad en todas y cada una de los actos hace nuestros clientes, proveedores y comunidad en general, imprimiendo un sentido de confianza, fiabilidad y transparencia en nuestro trabajo.

1.1.7. Estructura Organizacional.

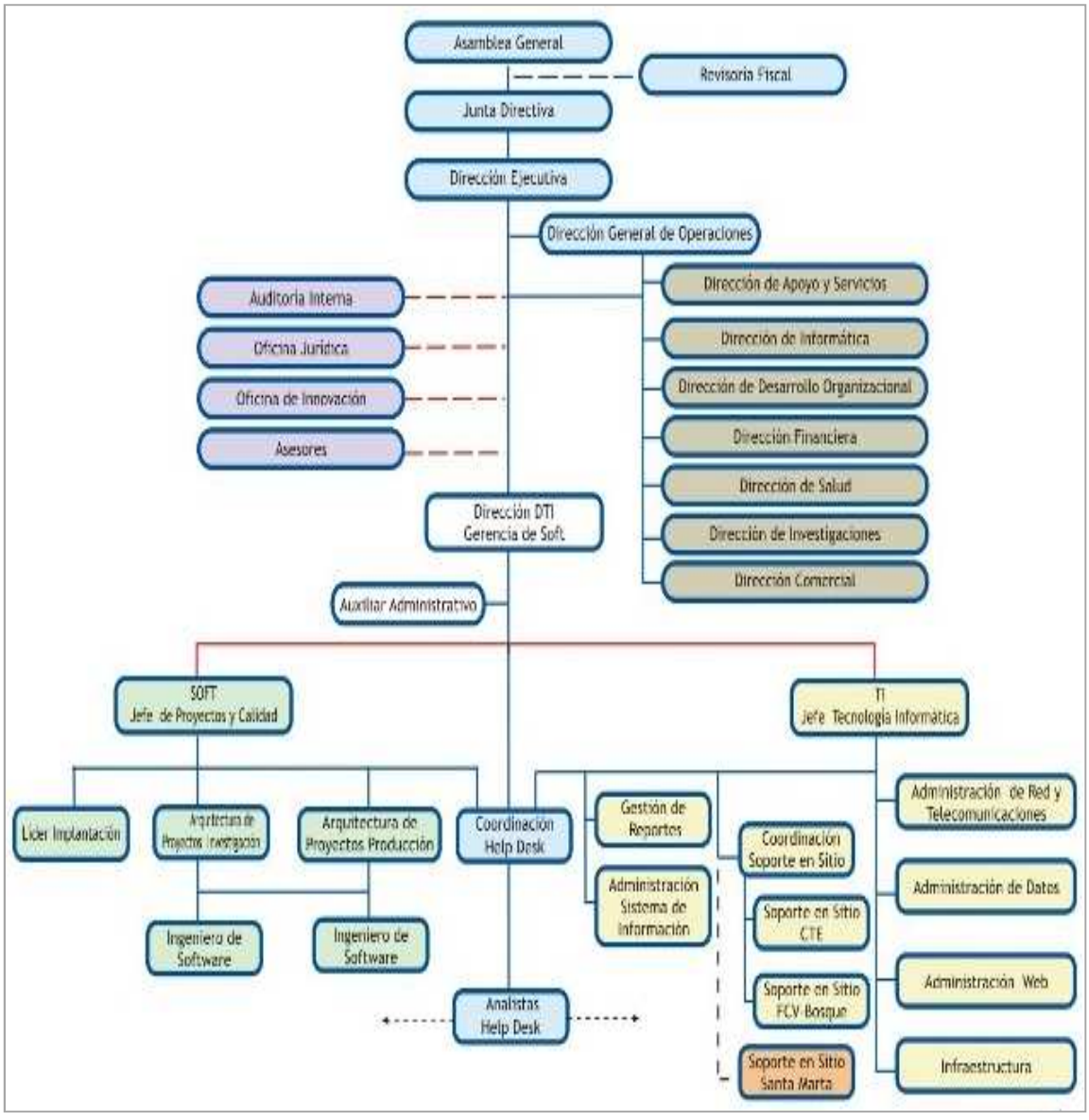

Figura 1. Organigrama FVC Soft

1.1.8. Responsabilidades a Cargo

1. Brindar asesoría y soporte técnico a los usuarios para la resolución de dudas y problemas relacionados con la aplicación.

2. Servir de soporte básico a los sistemas de información, entendiéndose por esto la solución a problemas de impresión, manipulación de archivos, estrategias de respaldo y recuperación y demás actividades directamente ligadas con la informática.

3. Diagnosticar e identificar errores de mal funcionamiento del software.

4. Identificar tendencias y tomar las acciones conducentes a remediar globalmente aquellos problemas que repetidamente son reportados y/o requeridos por su trabajo.

5. Realizar la respectiva documentación del desarrollo, según los estrictos modelos de calidad.

6. Controlar los documentos del sistema de calidad que estén bajo su responsabilidad participando activamente en la revisión de los mismos.

7. Generar y ejercer un control sobre registros de la aplicación que estén contemplados dentro de sus procesos y responsabilidades, asegurando su identificación, almacenamiento y protección.

8. Comunicar y reportar cualquier eventualidad o situación que potencialmente pueda generar una inconformidad, siguiendo el proceso de acciones preventivas.

9. Aplicar las acciones necesarias para corregir las inconformidades que se puedan presentar, generando las respectivas acciones correctivas y planes de mejora que eviten posteriores ocurrencias.

1.2. DESCRIPCIÓN DEL PROYECTO

1.2.1. Planteamiento del problema

Actualmente la Fundación Cardiovascular cuenta con el sistema de información (Cliente-Servidor) SAHI como principal sistema de gestión, este sistema se encarga en labores tanto financieros, como de administración medica, la UEN Telemedicina brinda servicios médicos por medio de SAHI y de un portal Web propio. Dicha situación ha generado procesos arduos y un sinfín de inconvenientes a los usuarios médicos, ya que la información con la cual se desarrollan y brindan sus servicios dichas aplicaciones, yacen de lugares diferentes, es decir, tienen bases de datos diferentes.

El problema crece cuando para desarrollar ciertos procesos del portal Web, se necesita información del sistema de información SAHI, lo cual obliga a los usuarios a salir y a entrar de una aplicación a otra, con diferentes nombres de usuario y claves.

Ante lo anterior, se planeo estratégicamente una unificación de aplicativos que se manipularan por medio de un portal Web, buscando la unificación de la información. Y por consiguiente brindar un mejor servicio médico, más confiable y de la más alta calidad.

La idea es ofrecer un entrenamiento eficaz a los médicos y que éstos a su vez, cuenten con información de calidad que les permita plantear casos con menos margen de error.

De igual forma, un médico que esté realizando rotación por fuera de la ciudad, puede tomar exámenes en línea a la hora que prefiera sin tener que trasladarse y puede comunicarse con pares de todo el mundo para consultar casos y diagnósticos. Se espera que en un futuro puedan también intercambiar información a través del celular.

1.2.2. Objetivos

1.2.2.1. Objetivo General

Diseñar, desarrollar e implementar un aplicativo en ambiente WEB, que sirva de plataforma interactiva, para facilitar a la comunidad médica a la hora de

diagnosticar, compartir información entre los usuarios (médicos), controlar historiales clínicos, observar el progreso de un paciente, registrar eventos y anomalías. En tiempo diferido y real (Chat), para la optima atención de los pacientes controlados por las secciones TELEMEDICINA, Centro Emisor y Centro de Referencia.

1.2.2.2. Objetivos específicos:

Ajustar de una manera amigable para el usuario, toda la información acerca de las aplicaciones ya existentes que se van a unificar en el portal interactivo (Telemedicina, Tele-UCI, Tele-Consulta, Tele-Apoyo, Diagnostico, Monitoreo, Chat, Video paciente).

Facilitar la comunicación entre los distintos usuarios del sistema, utilizando los diferentes estándares a la hora de diagnosticar a un paciente por medio de la WEB.

Facilitar al usuario (comunidad médica) la consulta del historial clínico de cualquier paciente registrado en el sistema, siempre y cuando sea autorizado por el doctor, que se le asigno su cuidado.

Garantizar un elevado nivel de seguridad entre los diferentes usuarios del sistema, manteniendo una estricta integridad y seguridad de la información a transmitir.

Diseñar la aplicación que se desenvuelva sin ningún tipo de inconvenientes en cualquier navegador WEB.

Controlar la administración de los perfiles de la comunidad médica (ya sean enfermeros, doctores, agentes del Contact Center, etc.).

1.2.3. Justificación

Para el óptimo desarrollo y avance de la medicina moderna en Colombia y en el mundo, es de vital importancia que el sector médico de toda una región o de todo un país esté en continua comunicación e interacción, en el control de historiales clínicos, reporte de anomalías u observaciones de un paciente, en tiempo real, para poder llevar a cabo las posibles correcciones o recomendaciones que puedan determinar los diferentes doctores de toda la región, el país o el mundo.

Para lograr lo anterior, se propone establecer un portal interactivo el cual pueda ser accedido desde cualquier parte del mundo, que cuente una conexión a internet, donde los doctores de los más distinguidos hospitales puedan controlar, interactuar y enterarse de los diferentes reportes, acerca del estado de los pacientes que se encuentren en otras regiones del país o del mundo.

1.2.4. Impacto

- **Técnico**

Teniendo en cuenta los aspectos mencionados anteriormente que hacen referencia al control del estado de pacientes y la disposición de la empresa en buscar nuevas y mejores alternativas para seguir avanzando en el sector médico, se hace la presentación de este proyecto, el cual busca que a través del diseño e implementación de un portal interactivo, proporcionarle a la Fundación Cardiovascular un avance en su proceso de evolución hacia las nuevas tendencias de desarrollo de software.

- **Social**

El desarrollo de este portal web se enfoca en la unificación de diferentes aplicaciones que dependen entre sí, con el fin de mejorar la calidad en la atención y control de los pacientes de la Fundación Cardiovascular, así como también su confiabilidad en el manejo de la información.

Dado que la información para el control de pacientes existente se manipula desde aplicaciones diferentes, se encuentra en uso una aplicación cliente-servidor desde la cual se maneja la administración medica de historias clínicas y una aplicación web encargada del servicio de tele consultas (chat) y envío de estudios para lectura (correo). Con el presente proyecto se busca unificar todas las aplicaciones existentes, con el fin de crear un portal web diseñado e implementado en Java con FrameWork Struts y con un Sistema Relacional de Base de Datos en SQL2005 en donde se mejorará notablemente el rendimiento en las transacciones y los novedosos diseños en la nueva plataforma web.

1.2.5. Viabilidad:

- **Técnica**

La Fundación Cardiovascular en su larga trayectoria, posee el equipo de hardware y software necesario para el desarrollo e implantación de la aplicación web propuesta y cuenta también con profesionales calificados y de experiencia en el manejo de herramientas de desarrollo de software.

- **Económica**

La empresa no tendrá que invertir en la compra de equipos especializados para el desarrollo del proyecto, puesto que cuenta con computadores actualizados. Igualmente no tendrá que involucrarse en rigurosos pagos a personal nuevo para el desarrollo porque ella cuenta con su propio grupo de programadores.

- **Social**

El desarrollo de este proyecto concuerda con la visión de la Fundación Cardiovascular, puesto que la tendencia es modernizar y actualizar continuamente sus servicios y productos, que son los puntos claves que la distinguen sobre otras empresas.

1.2.6. Cronograma de Actividades

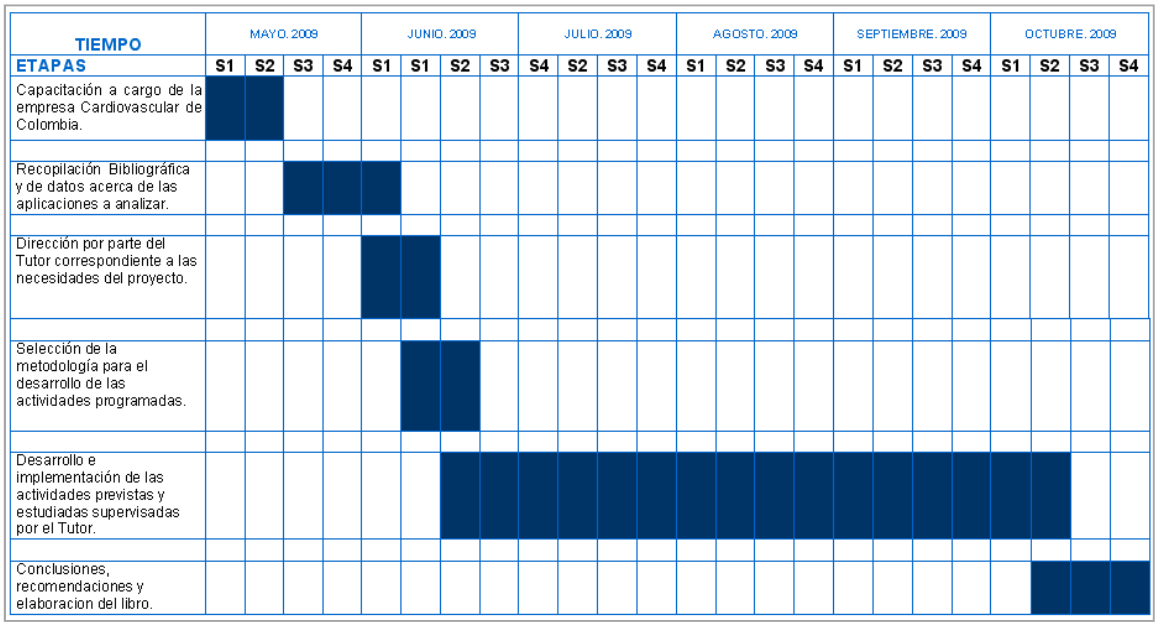

Figura 2. Cronograma de la práctica empresarial

2. MARCO TEÓRICO

Se pretende enmarcar dentro de este capítulo las herramientas teóricas utilizadas para el desarrollo de la práctica, haciendo Descripción es detallada del funcionamiento y la estructura de la metodología utilizada.

2.1. FUNDAMENTACION METODOLOGÍA

2.1.1. Metodologías Utilizadas en el desarrollo del software

Busca mostrar las principales ventajas y desventajas de algunas de las principales metodologías de desarrollo utilizadas actualmente, así como su funcionamiento.

Modelo en Cascada

El modelo en cascada fue uno de los primeros publicados y ha sido la base para otros modelos y procesos de la ingeniería del software actual; consiste en tomar actividades fundamentales del proceso de análisis de requisitos y requerimientos, diseño, desarrollo o construcción, implantación, pruebas y mantenimiento; representándolas como fases separadas del proceso.

El modelo cascada contempla las siguientes fases:

Fase de análisis de requerimientos y especificación de requisitos: busca definir detalladamente los requerimientos de software y hardware del sistema, así como los servicios, restricciones, y objetivos, establecidos con los usuarios del sistema; en esta fase se realiza un diseño o bosquejo preliminar de la aplicación, y se busca detectar todas las posibles causas de error y se planifica su corrección.

Fase de diseño: en esta fase se establece toda la arquitectura del sistema, se identifican y describen abstracciones de todos los procesos y procedimientos, al igual que las relaciones entre los diversos componentes del mismo.

Fase de desarrollo o construcción: en esta fase se lleva a cabo la construcción y codificación de los módulos y unidades de software, además se realizan las pruebas unitarias a cada uno de los módulos por separado.

Pruebas e implantación: se realiza el acoplamiento de los distintos componentes del sistema, se realizan pruebas de funcionamiento en conjunto y se le entrega al cliente la solución ya probada y lista para su funcionamiento.

Operación y mantenimiento: el sistema es puesto en marcha con datos reales y funcionamiento real, se realiza la corrección de errores descubiertos y mejoras de último minuto, además se identifican nuevos requisitos.

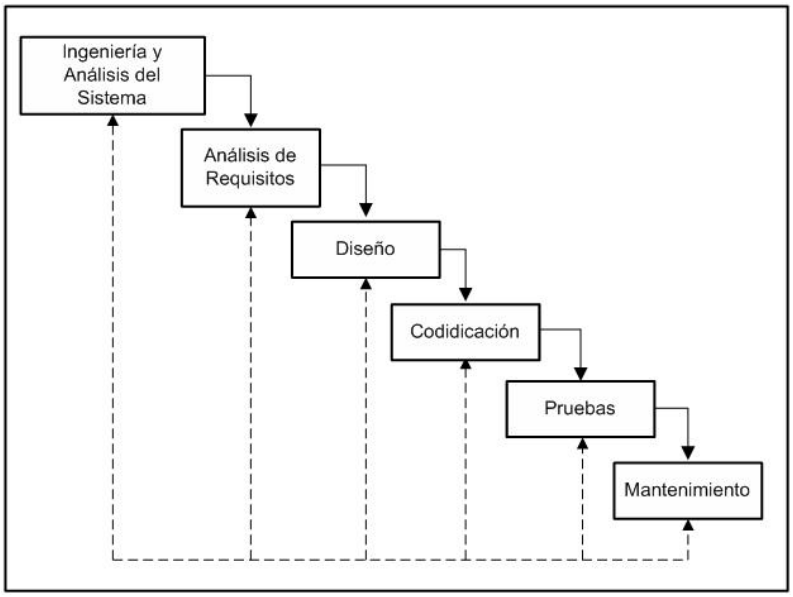

Figura 3. Diagrama de modelo en cascada

En la figura 3 se puede observar la interacción existente entre cada una de las fases del modelo, hay que notar que una fase no puede iniciar hasta que no haya sido completamente terminada la anterior, y por lo general se incluye la corrección de problemas encontrados en fases anteriores, y en cada fase se generan documentos que deben ser aprobados por el usuario.

En la práctica este modelo no es lineal se requieren varias iteraciones y se regresa varias veces a las fases previas, las principales ventajas de este modelo son:

• Una fase no comienza hasta que no hayan terminado las anteriores, esto supone que cada fase ha sido totalmente terminada y que la aparición de errores debe ser mínima

• Esta metodología fue base para las subsecuentes y fue muy aplicada por varias compañías durante más de dos décadas.

• Ayuda a prevenir el sobrepaso en las fechas de entrega y controla sobre costos.

Los principales problemas que se presentan con esta metodología son:

• Las iteraciones son costosas en tiempo y cantidad de trabajo, pues implican rehacer debido a la producción y aprobación de documentos.

• Debido al largo tiempo de espera para la entrega del proyecto es altamente probable que el software ya no cumpla con los requerimientos del usuario final.

• Como los problemas que se presentan son relegados para su posterior resolución, es probable que estos sean olvidados o se solucionen de una manera poco ortodoxa.

• El cliente pocas veces tiene claros todos los requisitos o se confunde y solicita cosas que en realidad no necesita.

2.2. APLICACIONES WEB

En sus inicios las páginas Web eran simplemente documentos estáticos que mostraban al usuario una cantidad de información sobre un tema determinado, pero que no permitían la interacción del usuario con esta información o con información nueva que él mismo necesitara, actualmente con la necesidad imperiosa de tener la información disponible desde cualquier lugar del mundo y poder interactuar con ella ha generado una gran demanda de tener aplicaciones robustas, rápidas fuertes y que permitan un manejo fiable de la información desde cualquier lugar del mundo.

Las aplicaciones Web se han convertido en pocos años en complejos sistemas con interfaces de usuario cada vez más parecidas a las aplicaciones de escritorio, dando servicio a procesos de negocio de considerable envergadura y estableciéndose sobre ellas requisitos estrictos de accesibilidad y respuesta. Esto ha exigido reflexiones sobre la mejor arquitectura y las técnicas de diseño más adecuadas.

2.2.1. Funcionamiento.

En los últimos años, la rápida expansión de Internet y del uso de intranets corporativas ha supuesto una transformación en las necesidades de información de las organizaciones. En particular esto afecta a la necesidad de que la información sea accesible desde cualquier lugar dentro de la organización e incluso desde el exterior. Asimismo, esta información debe ser compartida entre todas las partes interesadas, de manera que todas tengan acceso a la información completa (o a aquella parte que les corresponda según su función) en cada momento.

Estas necesidades han provocado un movimiento creciente de cambio de las aplicaciones tradicionales de escritorio hacia las aplicaciones Web, que por su idiosincrasia, cumplen a la perfección con las necesidades mencionadas anteriormente. Por tanto, los sitios Web tradicionales que se limitaban a mostrar información se han convertido en aplicaciones capaces de una interacción más o menos sofisticada con el usuario. Inevitablemente, esto ha provocado un aumento progresivo de la complejidad de estos sistemas y, por ende, la necesidad de buscar opciones de diseño nuevas que permitan dar con la arquitectura óptima que facilite la construcción de los mismos.

Una ventaja significativa en la construcción de aplicaciones Web que soporten las características de los browsers estándar es que deberían funcionar igual independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, Mac OS X, GNU/Linux, y otros sistemas operativos, la aplicación es escrita una vez y es mostrada casi en todos lados. Sin embargo, aplicaciones inconsistentes de HTML, CSS, DOM y otras especificaciones de browsers pueden causar problemas en el desarrollo y soporte de aplicaciones Web. Adicionalmente, la habilidad de los usuarios a personalizar muchas de las características de la interfaz (como tamaño y color de fuentes, tipos de fuentes, inhabilitar Javascript) puede interferir con la consistencia de la aplicación Web.

Otra aproximación es utilizar *Macromedia Flash* o *Java applets* para producir parte o toda la interfaz de usuario. Como casi todos los browsers incluyen soporte para estas tecnologías (usualmente por medio de plug-ins), aplicaciones basadas en Flash o Java pueden ser implementadas con aproximadamente la misma facilidad. Como hacen caso omiso de las configuraciones de los browsers estas tecnologías permiten más control sobre la interfaz, aunque incompatibilidad entre

implementaciones de Flash o Java puedan traer nuevas complicaciones. Por las similitudes con una arquitectura cliente-servidor, con un cliente un poco "especializado", hay disputas sobre si llamar a estos sistemas "aplicaciones Web"; un término alternativo es "aplicación enriquecida de Internet".

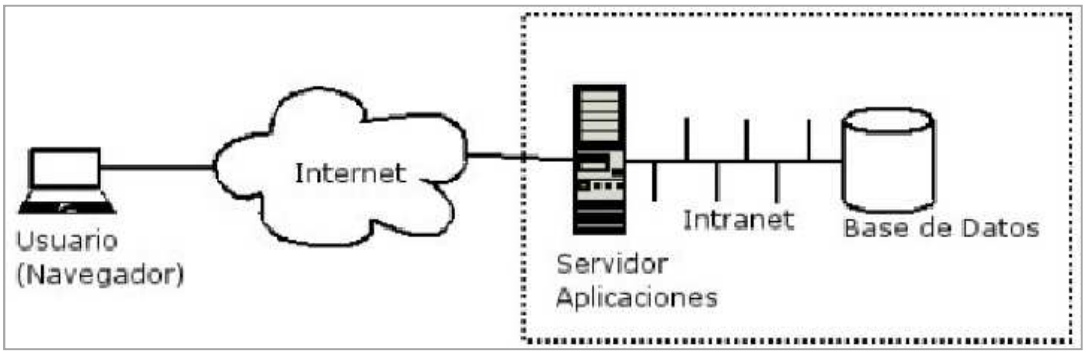

Figura 4. Funcionamiento de Aplicaciones Web

2.2.2. Arquitectura

El usuario interacciona con las aplicaciones Web a través del navegador. Como consecuencia de la actividad del usuario, se envían peticiones al servidor, donde se aloja la aplicación y que normalmente hace uso de una base de datos que almacena toda la información relacionada con la misma. El servidor procesa la petición y devuelve la respuesta al navegador que la presenta al usuario. Por tanto, el sistema se distribuye en tres componentes: el navegador, que presenta la interfaz al usuario; la aplicación, que se encarga de realizar las operaciones necesarias según las acciones llevadas a cabo por éste y la base de datos, donde la información relacionada con la aplicación se hace persistente. Esta distribución se conoce como el modelo o arquitectura de tres capas.

En la mayoría de los casos, el navegador suele ser un mero presentador de información (modelo de cliente delgado), y no lleva a cabo ningún procesamiento relacionado con la lógica de negocio. No obstante, con la utilización de applets, código javascript y DHTML la mayoría de los sistemas se sitúan en un punto intermedio entre un modelo de cliente delgado y un modelo de cliente grueso (donde el cliente realiza el procesamiento de la información y el servidor sólo es responsable de la administración de datos). No obstante el procesamiento realizado en el cliente suele estar relacionado con aspectos de la interfaz (como

ocultar o mostrar secciones de la página en función de determinados eventos) y nunca con la lógica de negocio.

En todos los sistemas de este tipo y ortogonalmente a cada una de las capas de despliegue comentadas, podemos dividir la aplicación en tres áreas o niveles:

• **Nivel de presentación**: es el encargado de generar la interfaz de usuario en función de las acciones llevadas a cabo por el mismo.

• **Nivel de negocio**: contiene toda la lógica que modela los procesos de negocio y es donde se realiza todo el procesamiento necesario para atender a las peticiones del usuario.

• **Nivel de administración de datos**: encargado de hacer persistente toda la información, suministra y almacena información para el nivel de negocio.

Los dos primeros y una parte del tercero (el código encargado de las actualizaciones y consultas), suelen estar en el servidor mientras que la parte restante del tercer nivel se sitúa en la base de datos (notar que, debido al uso de procedimientos almacenados en la base de datos, una parte del segundo nivel también puede encontrarse en la misma). Teniendo en cuenta estas características en la arquitectura de los sistemas Web, a continuación veremos algunos patrones de diseño de aplicación básica que pueden facilitar un diseño apropiado para este tipo de sistemas.

Uno de los patrones que ha demostrado ser fundamental a la hora de diseñar aplicaciones Web es el **Modelo-Vista-Control (MVC)**. Este patrón propone la separación en distintos componentes de la interfaz de usuario (vistas), el modelo de negocio y la lógica de control. Una vista es una “fotografía” del modelo (o una parte del mismo) en un determinado momento. Un control recibe un evento disparado por el usuario a través de la interfaz, accede al modelo de manera adecuada a la acción realizada, y presenta en una nueva vista el resultado de dicha acción. Por su parte, el modelo consiste en el conjunto de objetos que modelan los procesos de negocio que se realizan a través del sistema. En una aplicación Web, las vistas serían las páginas HTML que el usuario visualiza en el navegador. A través de estas páginas el usuario interactúa con la aplicación, enviando eventos al servidor a través de peticiones HTTP.

En el servidor se encuentra el código de control para estos eventos, que en función del evento concreto actúa sobre el modelo convenientemente. Los

resultados de la acción se devuelven al usuario en forma de página HTML mediante la respuesta HTTP.

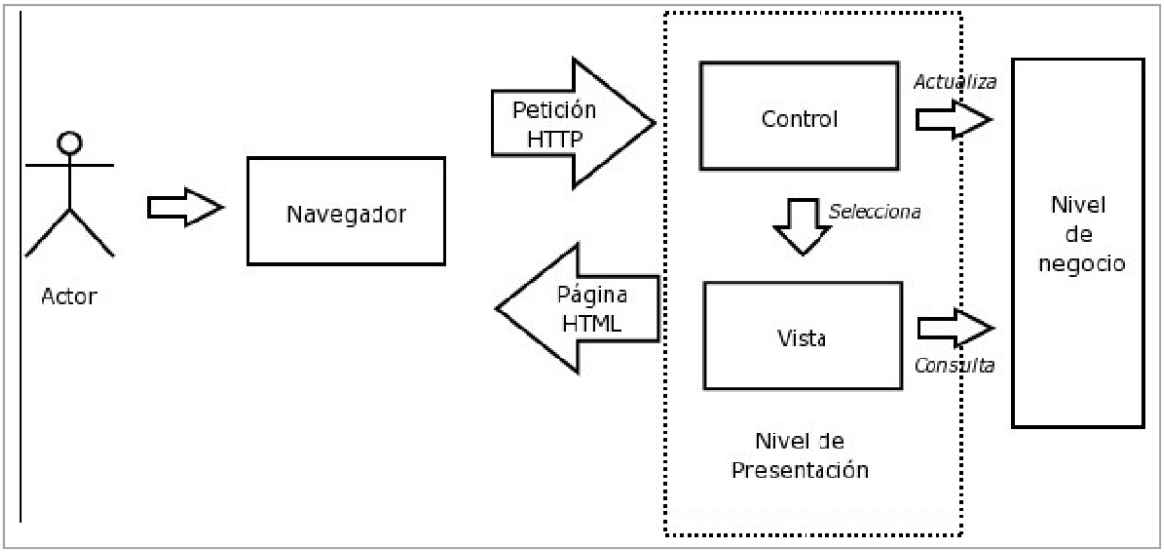

Figura 5. Representación del modelo tres capas

La clave está en la separación entre vista y modelo. El modelo suele ser más estable a lo largo del tiempo y menos sujeto a variaciones mientras que las vistas puede cambiar con frecuencia, ya sea por cambio del medio de presentación (por ejemplo HTML a WAP o a PDF) o por necesidades de usabilidad de la interfaz o simple renovación de la estética de la aplicación. Con esta clara separación las vistas pueden cambiar sin afectar al modelo y viceversa. Los controladores son los encargados de hacer de puente entre ambos, determinando el flujo de salida de la aplicación (qué se ve en cada momento).

2.3. INTRODUCCIÓN A LOS FRAMEWORKS

2.3.1. Definición

Un framework, en el desarrollo de software, es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

- **Breve Introducción**

Son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Por ejemplo, un equipo que usa Apache Struts para desarrollar un sitio web de un banco, puede enfocarse en cómo los retiros de ahorros van a funcionar en lugar de preocuparse de cómo se controla la navegación entre las páginas en una forma libre de errores. Sin embargo, hay quejas comunes acerca de que el uso de frameworks añade código innecesario y que la preponderancia de frameworks competitivos y complementarios significa que el tiempo que se pasaba programando y diseñando ahora se gasta en aprender a usar frameworks.

Fuera de las aplicaciones en la informática, puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo. Es el esqueleto sobre el cual varios objetos son integrados para una solución dada.

Después de todo, un Framework no debe ser consciente de todos estos requerimientos sin tener que ser intrusivo con las aplicaciones que permite dentro de sí mismo. A esto le sumamos la capacidad de extenderse sin prejuicios para diversificar la expresión del programa mismo.

2.3.2. Arquitectura

Dentro de este aspecto, podemos basarnos en el modelo MVC (Controlador => Modelo => Vista) ya que debemos fragmentar nuestra programación. Tenemos que contemplar estos aspectos básicos en cuanto a la implementación de nuestro sistema:

- **Controlador**:

Con este apartado podemos controlar el acceso (incluso todo) a nuestra aplicación, esto pueden ser: archivos, scripts o programas; cualquier tipo de

información que permita la interfaz. Así, podremos diversificar nuestro contenido de forma dinámica, y estática (a la vez); pues, sólo debemos controlar ciertos aspectos (como se ha mencionado antes).

- **Modelo**:

Este miembro del controlador maneja las operaciones lógicas, y de manejo de información (previamente enviada por su ancestro) para resultar de una forma explicable, y sin titubeos. Cada miembro debe ser meticulosamente llamado, en su correcto nombre y en principio, con su verdadera naturaleza: el manejo de información, su complementación directa.

- **Vista**:

Al final, a este miembro de la familia le corresponde dibujar, o expresar la última forma de los datos: la interfaz gráfica que interactúa con el usuario final del programa (GUI). Después de todo, a este miembro le toca evidenciar la información obtenida hasta hacerla llegar con el controlador. Solo (e inicialmente), nos espera demostrar la información.

2.3.3. Estructura

Dentro del controlador, modelo o vista podemos manejar lo siguiente: datos. Depende de nosotros como interpretar y manejar estos 'datos'. Ahora, sabemos que el único dato de una dirección estática web es: conseguir un archivo físico en el disco duro o de internet, etc. e interpretado o no, el servidor responde.

El modelo, al igual que el controlador y la vista, maneja todos los datos que se relacionen consigo (solo es el proceso medio de la separación por capas que ofrece la arquitectura MVC). Y sólo la vista, puede demostrar dicha información. Con lo cual ya hemos generado la jerarquía de nuestro programa: Controlador, Modelo y Vista.

- **Lógica**

Al parecer, debemos inyectar ciertos objetos dentro de sus parientes en esta aplicación, solo así compartirán herencia y coherencia en su aplicación.

Rápidamente, para una aplicación web sencilla debemos establecer estos objetos:

- Una base (MVC)
    - Controlador: éste debe ser capaz de manejar rutas, archivos, clases, métodos y funciones.
    - Modelo: es como un script habitual en el servidor, solo que agrupado bajo un 'modelo' reutilizable.
    - Vista: como incluyendo cualquier archivo en nuestra ejecución, muy simple.

- Un sistema
    - Ruteado: con él podemos dividir nuestras peticiones sin tantas condicionales.
    - Cargador.

2.3.4. Ventajas y Desventajas.

Ventajas:

- El programador no necesita plantearse una estructura global de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar".

- Facilita la colaboración. Cualquiera que haya tenido que "pelearse" con el código fuente de otro programador (¡o incluso con el propio, pasado algún tiempo!) sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.

- Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al framework concreto para facilitar el desarrollo.

- Otra ventaja de los frameworks, y en especial de esta acepción amplia, es la portabilidad de aplicaciones de una arquitectura a otra. Por ejemplo, los bytecodes generados a partir del código fuente de clases en Java pueden ser ejecutados sobre cualquier máquina virtual, independientemente de la arquitectura hardware y software subyacente.

Desventajas:

- El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.

- La reutilización de componentes software al por mayor. Los frameworks son los paradigmas de la reutilización.
- La dependencia del código fuente de una aplicación con respecto al framework. Si se desea cambiar de framework, la mayor parte del código debe reescribirse.

- La demanda de grandes cantidades de recursos computacionales debido a que la característica de reutilización de los frameworks tiende a generalizar la funcionalidad de los componentes. El resultado es que se incluyen características que están "de más", provocando una sobrecarga de recursos que se hace más grande en cuanto más amplio es el campo de *reutilización.*

2.4. ARQUITECTURA STRUTS

2.4.1. Definición

Es un framework de la capa de presentación que implementa el patrón Modelo Vista Controlador en Java.

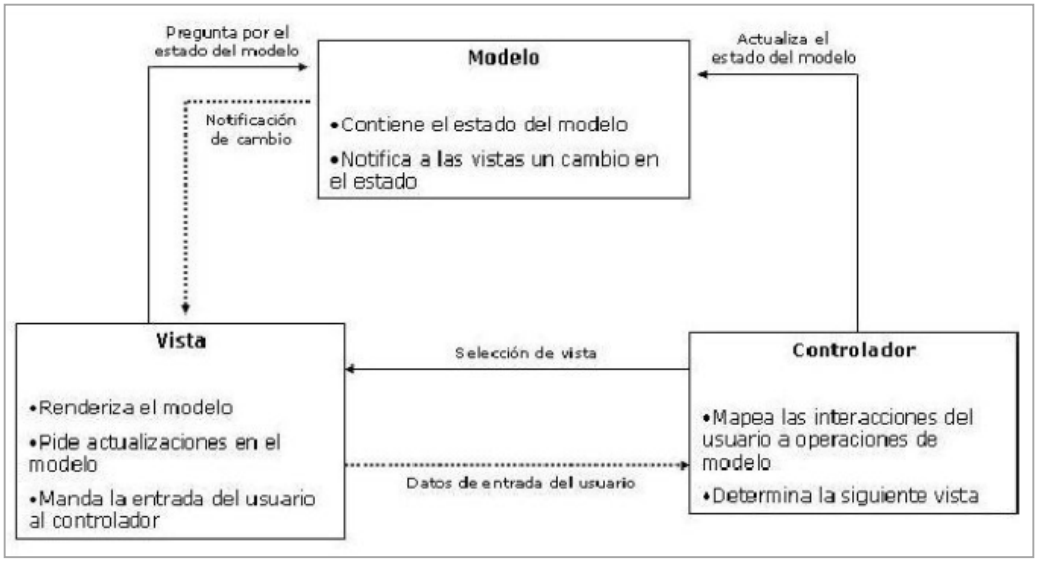

Figura 6. Patrón Modelo – Vista – Controlador

Evidentemente, como todo framework intenta simplificar 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 interfaz.

MVC (Modelo-Vista -Controlador) es un patrón de diseño aportado originariamente por el lenguaje SmallTalk a la Ingeniería del Software. Consiste principalmente en dividir las aplicaciones en tres partes:

- Controlador
- Modelo
- Vistas.

El controlador es el encargado de redirigir o asignar una aplicación a cada petición; el controlador debe poseer de algún modo, un "mapa" de correspondencias entre peticiones y respuestas que se les asignan.

El modelo seria la lógica de negocio a fin de cuentas. Una vez realizadas las operaciones necesarias el flujo vuelve al controlador y este devuelve los resultados a una vista asignada. El anterior gráfico nos muestra la interacción entre el Modelo, la Vista y el Controlador.

Anteriormente dijimos que Struts implementa el patrón MVC por ende nos debe proveer o dar accesibilidad a un Controlador, al Modelo y la Vista. 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. 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 a la hora de realizar los mantenimientos en el código.

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.

2.4.2. Funcionamiento en aplicaciones WEB

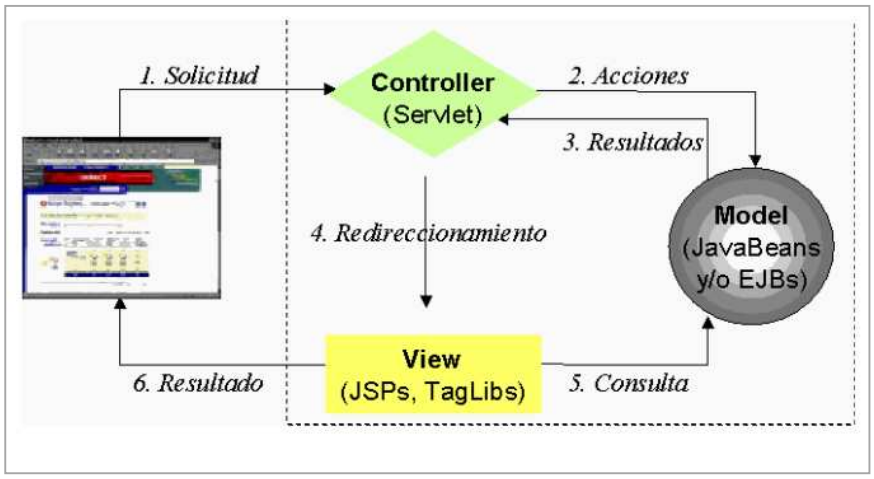

Figura 7. Funcionalidad de Struts.

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 JSPs, las cuales podrán consultar los objetos del Modelo para mostrar información de los mismos. Este gráfico nos provee una visión más detallada del funcionamiento de Struts:

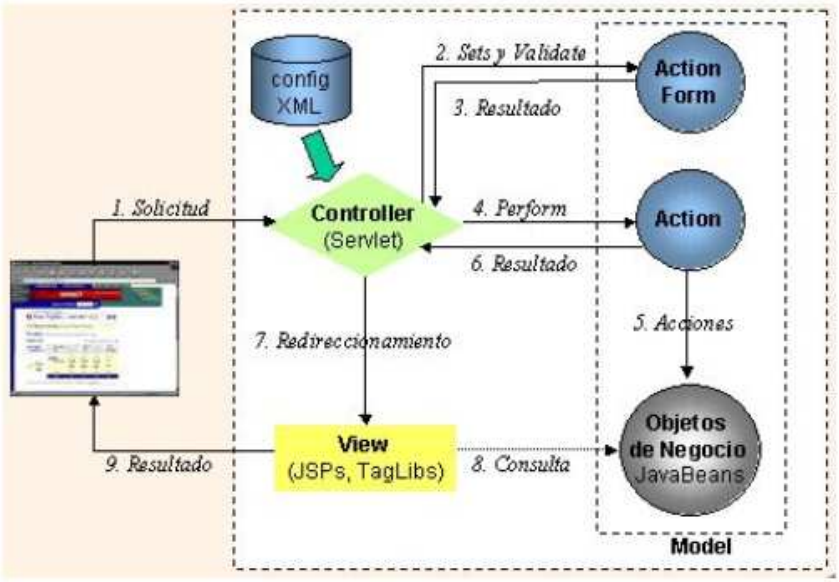

Figura 8. Funcionalidad detallada de Struts.

2.4.3. Ventajas y Desventajas

Ventajas de Struts

- El uso del mecanismo de tags JSP, promueve la reutilización de código y código abstracto de Java del archivo JSP. Esto permite una integración agradable en la herramienta de desarrollo basada en JSP que permite autoría con tags.

- Librería Tag, Struts provee un punto de partida si estás comenzando a aprender la tecnología de tags JSP tag.

- Open source, está disponible el código y todos pueden usar las revisiones del código.

- Implementación simple de MVC, Struts ofrece alguna comprensión si deseas crear tu propia implementación.

- Manejo del problema de espacio, Divide y vencerás es una forma agradable de resolver el problema y hacerlo manejable. Por su puesto, el problema se hace más complejo y necesita más manejo.

Desventajas de Struts

- Cambio, El framework está sufriendo una rápida cantidad de cambios. Han ocurrido una gran cantidad de cambios desde Struts 0.5 y 1.0. Se debe descargar la versión más reciente para evitar métodos desaprobados. Se deben modificar muchas veces los códigos a causa de los cambios en Struts.

- Nivel correcto de abstracción, Una de las librerías de tags de Struts es el Tag Logic, el cual maneja generación de condicionales de salida, pero no previene al desarrollador de los problemas con el código Java. Cual sea el tipo de framework que decidas usar, debes entender el entorno en el cual se despliega y se mantiene. Pero es más fácil decirlo que hacerlo.

- Libertad limitada, Struts está basado en una solución MVC, lo que significa ser implementado con HTML, archivos JSP, y servlets.

- Soporte de aplicaciones J2EE, Struts requiere un contenedor que soporte especificaciones JSP 1.1 y Servlet 2.2. Esto solo no resolverá los problemas de instalación, a menos que se utilice Tomcat 3.2.

- Complejidad, Separar el problema en partes introduce complejidad. Con los constantes cambios ocurriendo, esto puede ser frustrante a veces.

2.4.4. Herramienta de Desarrollo (IDE) MyEclipse

2.4.4.1. Descripción General

MyEclipse es un disponible en el comercio Empresa Java y AJAX IDE creado y mantenido por la compañía Genuitec, un miembro fundador del Fundación del eclipse.

MyEclipse se construye sobre Eclipse plataforma e integra el propietario y soluciones abiertas de la fuente en el ambiente del desarrollo.

MyEclipse tiene dos versiones primarias una edición profesional y estándar. La edición estándar agrega las herramientas de la base de datos, diseñador visual de la tela, herramientas de la persistencia, Resorte herramientas, Puntales y JSF el filetear, y un número de otras características al perfil básico del revelador de Java del eclipse.

MyEclipse es una herramienta del aprovisionamiento que mantiene perfiles del software del eclipse, incluyendo los que utilizan MyEclipse. Además, MyEclipse está ofreciendo una versión modificada para requisitos particulares para los productos de IBM, eso agrega la ayuda específica para Software racional y desarrollo de WebSphere.

2.5. ARQUITECTURA HIBERNATE

2.5.1. Definición

Hibernate es una herramienta para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

Hibernate es una herramienta ORM completa que ha conseguido en un tiempo record una excelente reputación en la comunidad de desarrollo posicionándose claramente como el producto OpenSource líder en este campo gracias a sus prestaciones, buena documentación y estabilidad. Es valorado por muchos incluso como solución superior a productos comerciales dentro de su enfoque, siendo una muestra clara de su reputación y soporte la reciente integración dentro del grupo JBoss que seguramente generará iniciativas muy interesantes para el uso de Hibernate dentro de este servidor de aplicaciones.

Se empezó a desarrollar hace algo más de 2 años por Gavin King siendo hoy Gavin y Christian Bauer los principales gestores de su desarrollo. Hibernate parte de una filosofía de mapear objetos Java "normales", también conocidos en la comunidad como "POJOs" (Plain Old Java Objects), no contempla la posibilidad de automatizar directamente la persistencia de Entity Beans tipo BMP (es decir, generar automáticamente este tipo de objetos), aunque aún así es posible combinar Hibernate con este tipo de beans utilizando los conocidos patrones para la delegación de persistencia en POJOs.

Una característica de la filosofía de diseño de Hibernate ha de ser destacada especialmente, dada su gran importancia: puede utilizar los objetos Java definidos por el usuario tal cual, es decir, no utiliza técnicas como generación de código a partir de descriptores del modelos de datos o manipulación de bytecodes en tiempo de compilación (técnica conocida por su amplio uso en JDO), ni obliga a implementar interfaces determinados, ni heredar de una superclase. Utiliza en vez de ello el mecanismo de reflexión de Java, característica que le permite un modelado iterativo fluido y natural basado en UML, un factor fundamental para lograr un trabajo ágil y productivo. Además abre las puertas a utilizar herencia en el modelo de datos (esta opción estaría limitada si una herramienta nos obliga a que los objetos de datos hereden de una superclase por no soportar Java herencia múltiple).

2.5.2. Características

- No intrusivo (estilo POJO).
- Muy buena documentación.
- Comunidad activa con muchos usuarios.

- Transacciones, caché, asociaciones, polimorfismo, herencia, lazy loading, persistencia transitiva, estrategias de fetching.

- Robusto lenguaje de consulta (HQL): subqueries, outer joins, ordering, proyección (report query), paginación.

- Fácil testeo.

- No es estándar.

2.5.3. Conceptos Básicos

Hibernate se distingue entre objetos tipo transient y tipo persistent, para almacenar y recuperar estos objetos de la base de datos, el desarrollador debe mantener una conversación con el motor de Hibernate mediante un objeto especial, quizás el concepto clave más importante dentro Hibernate, que es la Sesión (clase Session). Se puede equiparar a grandes rasgos al concepto de conexión de JDBC y cumple un papel muy parecido, es decir, sirve para delimitar una o varias operaciones relacionadas dentro de un proceso de negocio, demarcar una transacción y aporta algunos servicios adicionales como una caché de objetos para evitar interacciones innecesarias contra la BD.

En este sentido veremos que la clase Session ofrece métodos como save(Object object), createQuery(String queryString), beginTransaction(), close(), etc. para interactuar con la BD tal como estamos acostumbrados a hacerlo con una conexión JDBC (de hecho "envuelve" una conexión JDBC), pero con una diferencia: mayor simplicidad, es decir, guardar un objeto, por ejemplo, consiste en algo así como session.save(miObjecto), sin necesidad de especificar una sentencia SQL, y esto es sólo un ejemplo muy trivial en el que ganamos relativamente poco utilizando Hibernate.

Con esto volvemos a los conceptos de transient y persistent: los primeros son objetos que sólo existen en memoria y no en un almacén de datos (recuérdese en este sentido también el modificador transient de Java), en algunos casos, no serán almacenados jamás en la base de datos y en otros es un estado en el que se encuentran hasta ser almacenados en ella. Los segundos se caracterizan por haber sido ya almacenados y ser por tanto objetos persistentes. Dicho de otra: manera los objetos transient han sido instanciados por el desarrollador sin haberlos almacenado mediante una sesión, los objetos persistentes han sido

creados y almacenados en una sesión o bien devueltos en una consulta realizada con la sesión.

Igual que con las conexiones JDBC hemos de crear y cerrar sesiones, aunque no hay una relación 1:1 entre sesiones y conexiones, es decir, no tenemos que abrir y cerrar simultáneamente sesiones y conexiones JDBC, la política a seguir dependerá del contexto del proceso de negocio de cada situación dándonos Hibernate amplias posibilidades para la implementación de nuestras políticas (conexiones JDBC gestionadas por la aplicación, por Hibernate, por un posible servidor de aplicaciones, etc.), siendo solamente necesario en la práctica crear y cerrar explícitamente las sesiones de Hibernate.

Hemos visto que las sesiones son un concepto ligado a un proceso de negocio, por tanto es natural pensar que una sesión siempre va a pertenecer a un mismo thread de ejecución (el que pertenece a la ejecución de un método de negocio para un usuario o sistema externo concreto), aunque técnicamente se pueden compartir sesiones entre threads, esto no se debe hacer jamás por no ser una buena política de diseño y los consecuentes problemas que puede generar.

Es decir, en un entorno multiusuario y por tanto multithread habrá por tanto múltiples sesiones simultáneas, cada una perteneciente a su correspondientes thread y con su contexto de objetos en caché, transacciones, etc. Como tal no sorprende que las sesiones no son “thread-safe” y que la información vinculada a ella no sea visible para otras sesiones. Es también lógico que tenga que existir una “institución” superior para crear sesiones y realizar operaciones comunes a los diferentes threads como lo puede ser la gestión de una caché compartida entre threads o caché de segundo nivel. Este elemento es la clase SessionFactory y en ella podremos encontrar métodos como openSession()o evict(Class persistentClass).

Por fin tenemos que pensar también en qué sucede si en un entorno de múltiples hilos de ejecución la aplicación accede a un mismo objeto desde dos sesiones diferentes. Vimos que una instancia de un objeto persistente nunca es compartida por dos sesiones al contar cada una con su propio contexto para ello, de modo que existirán dos instancias dentro de la misma máquina virtual Java para un “mismo” objeto de datos, lo cual no lleva al concepto de **identidad**.

Hay que distinguir entre identidad de instancia Java, es decir: objeto1 == objeto2, identidad persistente: objeto1. getId().equals(objeto2.getId)) y la identidad a nivel de base de datos (claves primarias iguales). Por tanto, puede haber dentro de la

misma máquina virtual varios objetos con la misma identidad persistente, pero diferentes identidades como instancias por ser objetos de datos Java que representan la misma entidad persistente.

Normalmente la identidad persistente del objeto y la identidad de base de datos coincidirán, pero esto puede no ser así para lógicas de negocio muy particularidad. En todo caso, esta política depende del desarrollador que puede jugar, por ejemplo, con sobre escribir el método equals() para definir un comportamiento peculiar y utilizar claves especiales para la identidad de objetos diferenciándolas o relacionándolas con la propiedad del objeto utilizada como clave primaria para su persistencia en la base de datos.

2.5.4. Hibernate Query Language

Hibernate nos proporciona además un lenguaje para el manejo de 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 nos permite usar un lenguaje intermedio que según la base de datos que usemos y el dialecto que especifiquemos será traducido al SQL dependiente de cada base de datos de forma automática y transparente

2.5.5. Configuración en herramienta de desarrollo

Para utilizar Hibernate en una aplicación, es necesario conocer como configurarlo. Hibernate puede configurarse y ejecutarse en la mayoría de aplicaciones java y entornos de desarrollo. Generalmente, Hibernate se utiliza en aplicaciones cliente/servidor de dos y tres capas, desplegándose Hibernate únicamente en el servidor. Las aplicaciones cliente normalmente utilizan un navegador Web, pero las aplicaciones swing y AWT también son usuales. Aunque solamente vamos a ver como configurar Hibernate en un entorno no gestionado, es importante comprender la diferencia entre la configuración de Hibernate para entornos gestionados y no gestionados:

- **Entorno gestionado**:

Los pools de recursos tales como conexiones a la base de datos permiten establecer los límites de las transacciones y la seguridad se debe especificar de forma declarativa, es decir en sus metadatos. Un servidor de aplicaciones J2EE,

tal como JBoss, Bea WebLogic o IBM WebSphere implementan un entorno gestionado para Java.

- **Entorno no gestionado**:

Proporciona una gestión básica de la concurrencia a través de un pooling de threads. Un contenedor de servlets, como Tomcat proporciona un entorno de servidor no gestionado para aplicaciones Web Java. Una aplicación stand-alone también se considera como no gestionada. Los entornos no gestionados no proporcionan infraestructura para transacciones automáticas, gestiones de recursos, o seguridad. La propia aplicación es la que gestiona las conexiones con la base de datos y establece los límites de las transacciones.

Tanto en un entorno gestionado como en uno no gestionado, lo primero que debemos hacer es iniciar Hibernate. Para hacer esto debemos crear una Session Factory desde una Configuración.

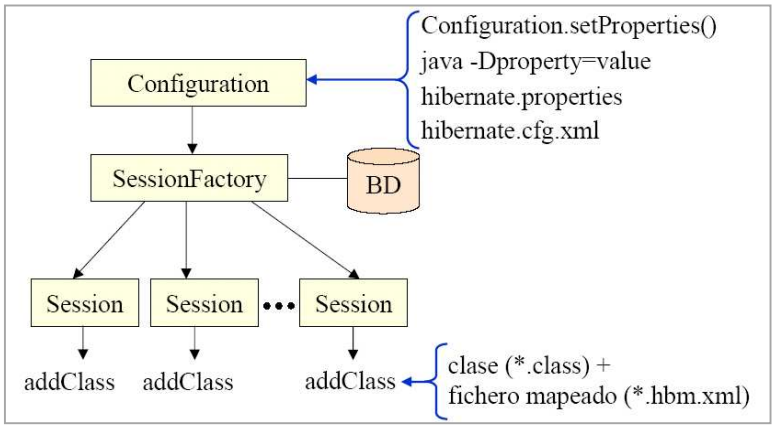

Figura 9. Esquema de configuración Hibernate

2.5.6. Configuración en la Base de Datos

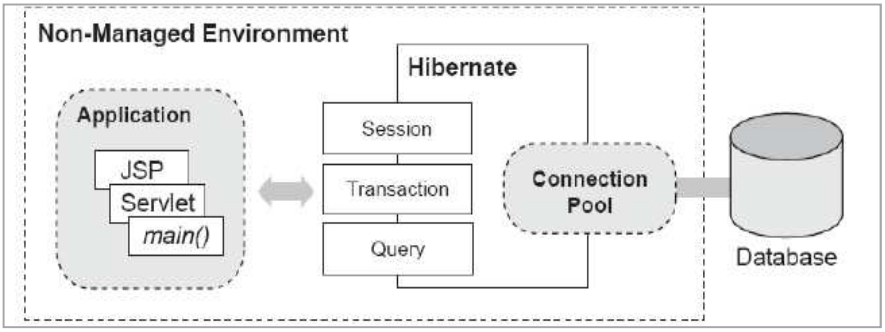

Figura 10. Esquema de configuración Hibernate 2.

Resumen de pasos de configuración

- Situar el *.jar del driver JDBC elegido y el fichero hibernate2.jar en nuestro classpath

- Añadir las dependencias de Hibernate (directorio (lib)) en el classpath. (lib/README.txt contiene una lista de librerías requeridas y opcionales).

- Elegir y configurar un pool de conexiones JDBC

- Determinar las propiedades de Configuración en un fichero hibernate.properties en el classpath.

- Crear una instancia de Configuración en nuestra aplicación y cargar los ficheros de mapeado XML utilizando addResource() o addClass().

- Obtener una SessionFactory a partir de Configuration llamando a BuildSessionFactory().

2.5.7. Estructura

El API de Hibernate es una arquitectura de dos capas (Capa de persistencia y Capa de Negocio).

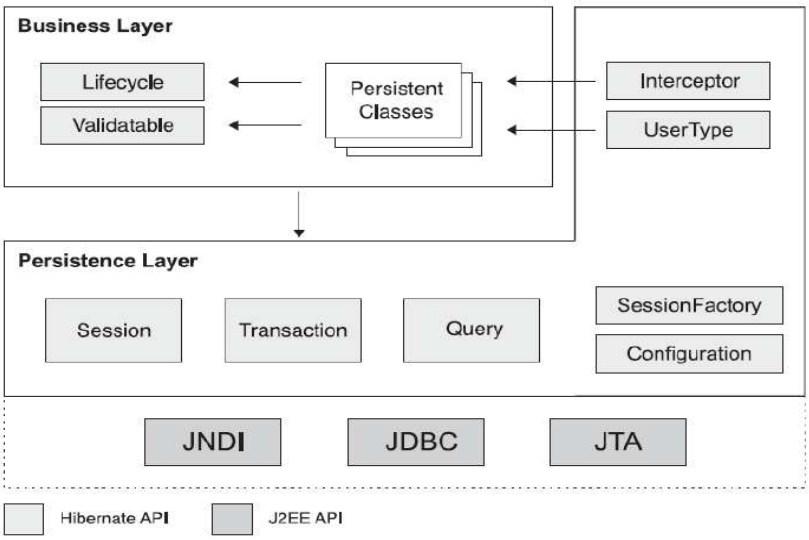

Figura 11. Roles de las interfaces Hibernate en las capas de persistencia

La Figura 11 muestra los roles de las interfaces Hibernate más importantes en las capas de persistencia y de negocio de una aplicación J2EE. La capa de negocio está situada sobre la capa de persistencia, debido a que actúa como un cliente de la capa de persistencia. Las Interfaces mostradas se clasifican de la siguiente forma:

- Interfaces llamadas por la aplicación para realizar operaciones básicas:
  - Session: interfaz primaria utilizada en cualquier aplicación Hibernate (SessionFactory).
  - Transaction

Query: Permite realizar peticiones a la base de datos y controla cómo se ejecuta dicha petición (query). Las peticiones se escriben en HQL o en el dialecto SQL nativo de la base de datos que estamos utilizando. Una instancia Query se utiliza para enlazar los parámetros de la petición, limitar el número de resultados devueltos por la petición y para ejecutar dicha petición.

- Interfaces llamadas por el código de la infraestructura de la aplicación para configurar Hibernate. La más importante es la clase **Configuration:** se utiliza para configurar y "arrancar" Hibernate. La aplicación utiliza una instancia de Configuration para especificar la ubicación de los documentos que indican el mapeado de los objetos y propiedades específicas de Hibernate, y a continuación crea la Session Factory.

- Interfaces callback que permiten a la aplicación reaccionar ante determinados eventos que ocurren dentro de la aplicación, tales como **Interceptor, Lifecycle, y Validatable.**

- Interfaces que permiten extender las funcionalidades de mapeado de Hibernate, como por ejemplo **UserType, CompositeUserType, e IdentifierGenerator.**

Además, Hibernate hace uso de APIs de Java, tales como JDBC, JTA (Java Transaction Api) y JNDI (Java Naming Directory Interface).

2.5.8. Ventajas

Si se está trabajando con programación orientada a objetos y bases de datos relacionales, seguramente habrás observado que estos son dos paradigmas diferentes.

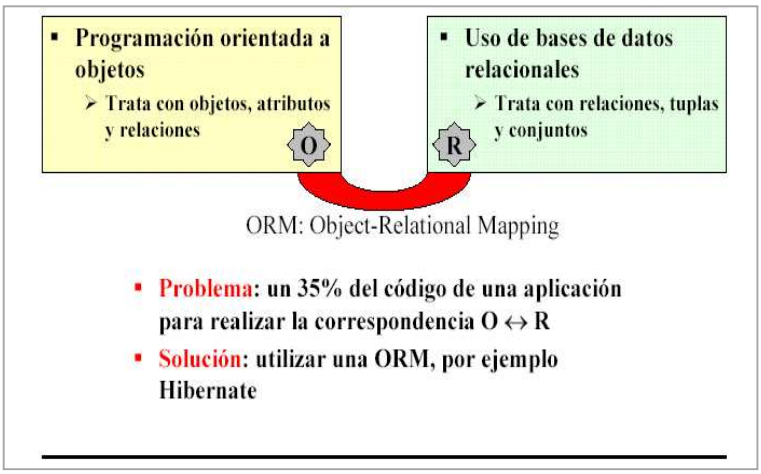

Figura 12. Programación orientada a objetos y bases de datos relacionales

El modelo relacional trata con relaciones, tuplas y conjuntos y es muy matemático por naturaleza. Sin embargo, el paradigma orientado a objetos trata con objetos, sus atributos y relaciones entre objetos. Cuando se quiere hacer que los objetos sean persistentes utilizando para ello una base de datos relacional, uno se da cuenta de que hay una desavenencia entre estos dos paradigmas, la también llamada diferencia objeto-relacional (object – relational gap"). Un mapeador objeto-relacional **(ORM)** nos ayudará a evitar esta diferencia.

Si estamos utilizando objetos en nuestra aplicación y en algún momento queremos que sean persistentes, normalmente abriremos una conexión JDBC, crearemos una sentencia SQL y copiaremos todos los valores de las propiedades sobre una PreparedStatement o en la cadena SQL que estemos construyendo. Esto podría ser fácil para un objeto de tipo valor (value object:VO) de pequeño tamaño pero consideremos esto para un objeto con muchas propiedades. Este no es el único problema.

Como se puede ver, la brecha existente entre los paradigmas de objeto y relacional se vuelve mucho mayor si disponemos de modelos con objetos "grandes". Y hay muchas más cosas a considerar como la carga lenta, las referencias circulares, el caché, etc. De hecho, hay estudios que demuestran que el 35% del código de una aplicación se produce como consecuencia del mapeado (correspondencia) entre los datos de la aplicación y el almacén de datos.

Entonces, lo que necesitamos es una herramienta **ORM** (Object Relational Mapping). Básicamente, una ORM intenta hacer todas estas tareas pesadas por nosotros. Con una buena ORM, sólo tendremos que definir la forma en la que establecemos la correspondencia entre las clases y las tablas una sola vez

(indicando que propiedad se corresponde con que columna, que clase con que tabla, etc.). Después de esto, podremos hacer cosas como utilizar **POJO'**s (Plain Old Java Objects) de nuestra aplicación y decirle a nuestra ORM que los haga persistentes, con una instrucción similar a esta: orm.save(myObject). Es decir, una herramienta puede leer o escribir en la base de datos utilizando VO's directamente.

Más formalmente: un modelo del dominio representa las entidades del negocio utilizadas en una aplicación Java. En una arquitectura de sistemas por capas, el modelo del dominio se utiliza para ejecutar la lógica del negocio (en Java, no en la base de datos). Esta capa del negocio se comunica con la capa de persistencia subyacente para recuperar y almacenar los objetos persistentes del modelo del dominio. ORM es el middleware en la capa de persistencia que gestiona la persistencia.

Hibernate es un ORM de libre distribución, que además, es de las más maduras y completas. Actualmente su uso está muy extendido y además está siendo desarrollada de forma muy activa. Una característica muy importante que distingue Hibernate de otras soluciones al problema de la persistencia, como los Ejes de entidad, es que la clase Hibernate persistente puede utilizarse en cualquier contexto de ejecución, es decir, no se necesita un contenedor especial para ello.
Utilizar un framework de ORM simplifica enormemente la programación de lógica de persistencia. Se trata de una idea completamente madura que cada vez se vuelve más popular. Nuestra lógica de negocios trabaja contra un modelo de dominio completamente orientado a objetos. Generamos entre un 30% y un 40% menos de código y el tipo de código generado es mucho más sencillo.

- Es fundamental conocer bien cómo funcionan las tecnologías que utilizamos. En el caso de Hibernate hemos visto que dependiendo de cómo hagamos las cosas puede afectar directamente al rendimiento de la aplicación.

- En cuanto al manejo de consultas Hibernate saca una ligera ventaja ya que tiene su propio lenguaje "HQL" que lo hace multi – motor de base de datos, eso es uno de los atractivos de Hibernate.

- Hibernate soporta la mayoría de los sistemas de bases de datos SQL. El Hibernate Query Language, diseñado como una extensión mínima, orientada a objetos, de SQL, proporciona un puente elegante entre los mundos objeto y relacional.

- Hibernate ofrece facilidades para recuperación y actualización de datos, control de transacciones, repositorios de conexiones a bases de datos, consultas programáticas y declarativas, y un control de relaciones de entidades declarativas.

- Hibernate es una muy buena herramienta, en lo que se refiere al “mapeo clases” en una base de datos relacional. Aunque, a la hora de referirse al manejo de transacciones y conexiones, le falta ser un poco más funcional.

- Hibernate es menos invasivo que otros marcos de trabajo de mapeo O/R. Se utilizan Reflection y la generación de bytecodes en tiempo de ejecución, y la generación del SQL ocurre en el momento de la arrancada. Esto nos permite desarrollar objetos persistentes siguiendo el lenguaje común de Java: incluyendo asociación, herencia, polimorfismo, composición y el marco de trabajo Collections de Java.

3. DESARROLLO DE LA PRÁCTICA

Este capítulo plantea el contexto y las actividades realizadas durante los seis meses de duración de la práctica empresarial, que permitieron cumplir con los objetivos a desarrollar, y describe el apoyo brindado a las diferentes etapas de los proyectos que en el momento se venían realizando dentro de la empresa.

Cabe resaltar la magnitud e importancia de un proyecto de este tipo ya que se pretende manejar en ambiente Web una aplicación donde la cantidad e transacciones es bastante grande y se maneja procesos que son de vital importancia para la empresa cliente (Telemedicina); lo cual implica un especial cuidado con el manejo y la seguridad de los datos pues la proyección del software es que funcione tanto en la intranet como desde Internet.

Esta práctica empresarial se baso en la continuación del desarrollo de una aplicación realizada por La Empresa “CreandoSoft” (Empresa externa), Este desarrollo estuvo orientado a satisfacer una serie de requerimientos solicitados por la UEN Telemedicina. Inicialmente se revisó la documentación del software en desarrollo y la fase de evolución de los proyectos que se estaban realizando en el momento, sobretodo de la aplicación principal que se estaba desarrollando, la cual consiste en el desarrollo de una aplicación Web para la unificación de las aplicaciones para el manejo de los servicios de control de pacientes de la Fundación Cardiovascular.

Es importante mencionar que se realizo una capacitación brindada por el tutor e Ingenieros y Diseñadores de Sistemas a cargo, que se oriento en el conocimiento las herramientas y conceptos del software a utilizar, como lo es el IDE Eclipse3.5 y diferentes FRAMEWORK como HIBERNATE, JSF y STRUTS.

3.1. DESCRIPCIÓN DE LA APLICACIÓN

3.1.1. Introducción

Antes de empezar a describir los procesos involucrados en la aplicación web (SAHIWEB) es prudente tener muy claro el concepto de Telemedicina, ya que los módulos a describir deben satisfacer en su totalidad la filosofía de telemedicina.

Definición telemedicina:

Es medicina practicada a distancia, que incluye diagnóstico, tratamiento y educación médica. Es un recurso tecnológico que posibilita la optimización de los servicios de atención en salud, ahorrando tiempo y dinero, facilitando el acceso a zonas distantes para tener la atención de especialistas. Así podemos definir los servicios, que la telemedicina presta:

- Servicios complementarios e instantáneos a la atención de un especialista. (Obtención de una segunda opinión)

- Diagnósticos inmediatos por parte de un médico especialista en un área determinada.

- Educación remota de alumnos de las escuelas de enfermería y medicina.

- Servicios de archivo digital de exámenes radiológicos, ecografías y otros.

Todo esto se traduce en una diminución de tiempos entre la toma de exámenes y la obtención de resultados, o entre la atención y el diagnostico certero del especialista, el cual no debe viajar o el paciente no tiene que ir a examinarse, reduciendo costos de tiempo y dinero.

La aplicación consta de cinco aplicaciones que anteriormente trabajaban por separado, que constituyen el manejo de los procesos fundamentales para el desarrollo de la Aplicación web (SAHIWEB).

3.2. Descripción por procesos.

3.2.1. Tele-Apoyo Diagnóstico o Estudios para Lectura

Este proceso de la aplicación SAHIWEB, consiste en brindar el consentimiento de un médico especialista (Consultor) ubicado en cualquier lado del país, con acceso a internet y registrado al sistema, a un médico con conocimientos médicos generales (Consultante) perteneciente a una entidad remisoria.

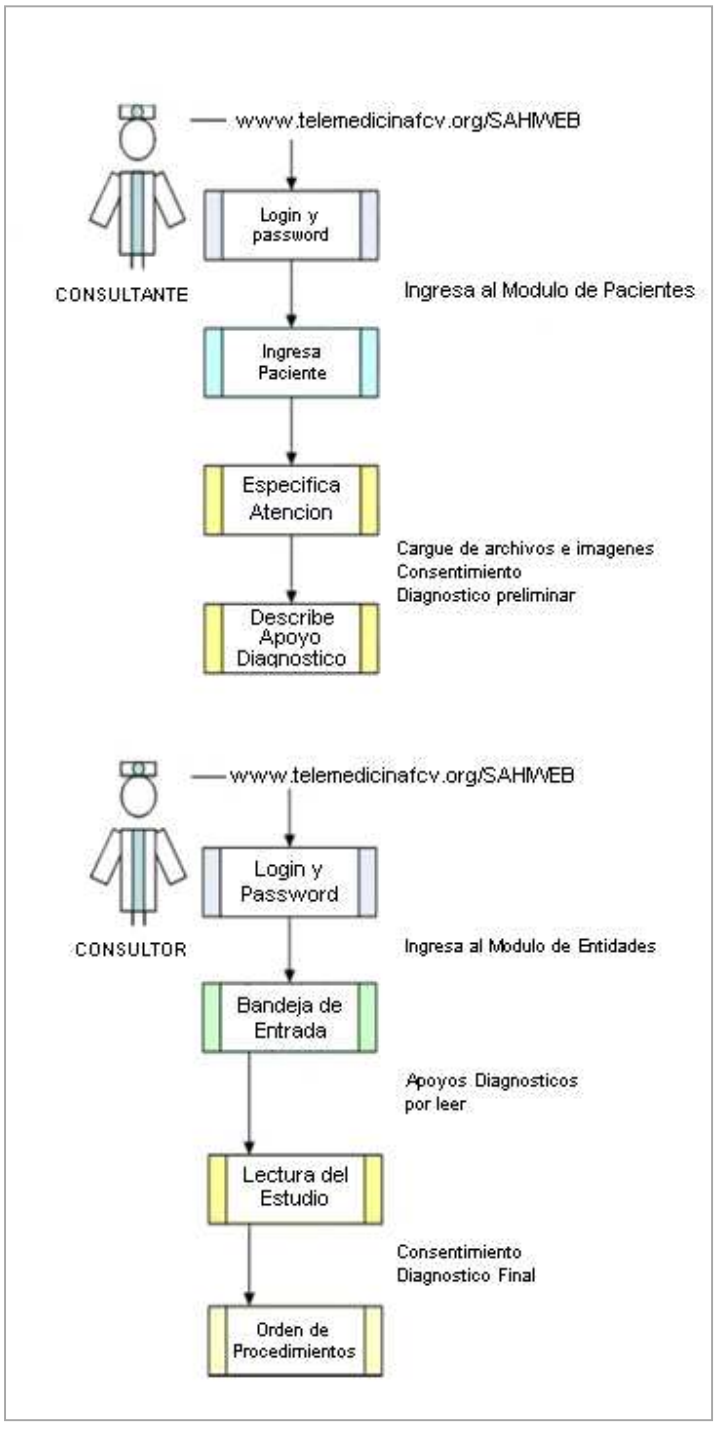

Figura 13. Diagrama de procesos de Tele-Apoyo Diagnostico

El soporte básicamente consiste en, rectificar o en algunas ocasiones recomendar el diagnóstico o diagnósticos preliminares asociados a la consulta, provenientes del médico consultante, lectura de radiografías y electrocardiogramas en tiempo diferido. Para explicar el proceso de una forma más informal, “la plataforma WEB

se comporta como un correo electrónico". Con la gran diferencia que se tiene un estricto control de la solicitud del servicio en trasfondo de la aplicación.

El médico consultante, escribe y carga archivos (archivos con extensión .jpg y .ecg) pertinentes a la consulta (Tele-Apoyo Diagnostico). El médico Especialista (consultor) recibe el (Tele-Apoyo Diagnostico) es notificado por la aplicación por medio de la bandeja de entrada cuando ingresa al portal.

Explicando el proceso de una manera más procedimental, cuando se presenta un caso médico particular, en algún hospital vinculado al sistema y el doctor consultante requiere la apreciación de un médico especialista, en base a exámenes concernientes a dicho caso, debe solicitar el servicio a través del portal. En pocas palabras un doctor que solicita el servicio de Tele-Apoyo Diagnostico sigue los siguientes pasos:

1. El doctor (medico Consultante) ingresa al portal WEB con su respectiva cuenta (login y password).

2. Comenta el caso y anexa la historia clínica, imágenes (Radiografías) o ECG (Electrocardiogramas). Escoge una especialidad a consultar y solo le queda esperar su respuesta.

3. Los doctores Especialistas matutinalmente (médicos consultores) revisan esta "bandeja de entrada" y contestan las respectivas ayudas diagnostica.

4. Una vez dada respuesta por el médico consultor, al médico consultante se le notificara la respuesta de la misma forma como se notifica una petición, en la misma "bandeja de entrada".

3.2.2. Tele-Consulta

Este proceso de la aplicación SAHIWEB, al igual que en Tele-Apoyo Diagnostico consiste en brindar el consentimiento de un médico especialista (Consultor) ubicado en cualquier lado del país, con acceso a internet y registrado al sistema, a un médico con conocimientos médicos generales (Consultante) perteneciente a una entidad remisoria.

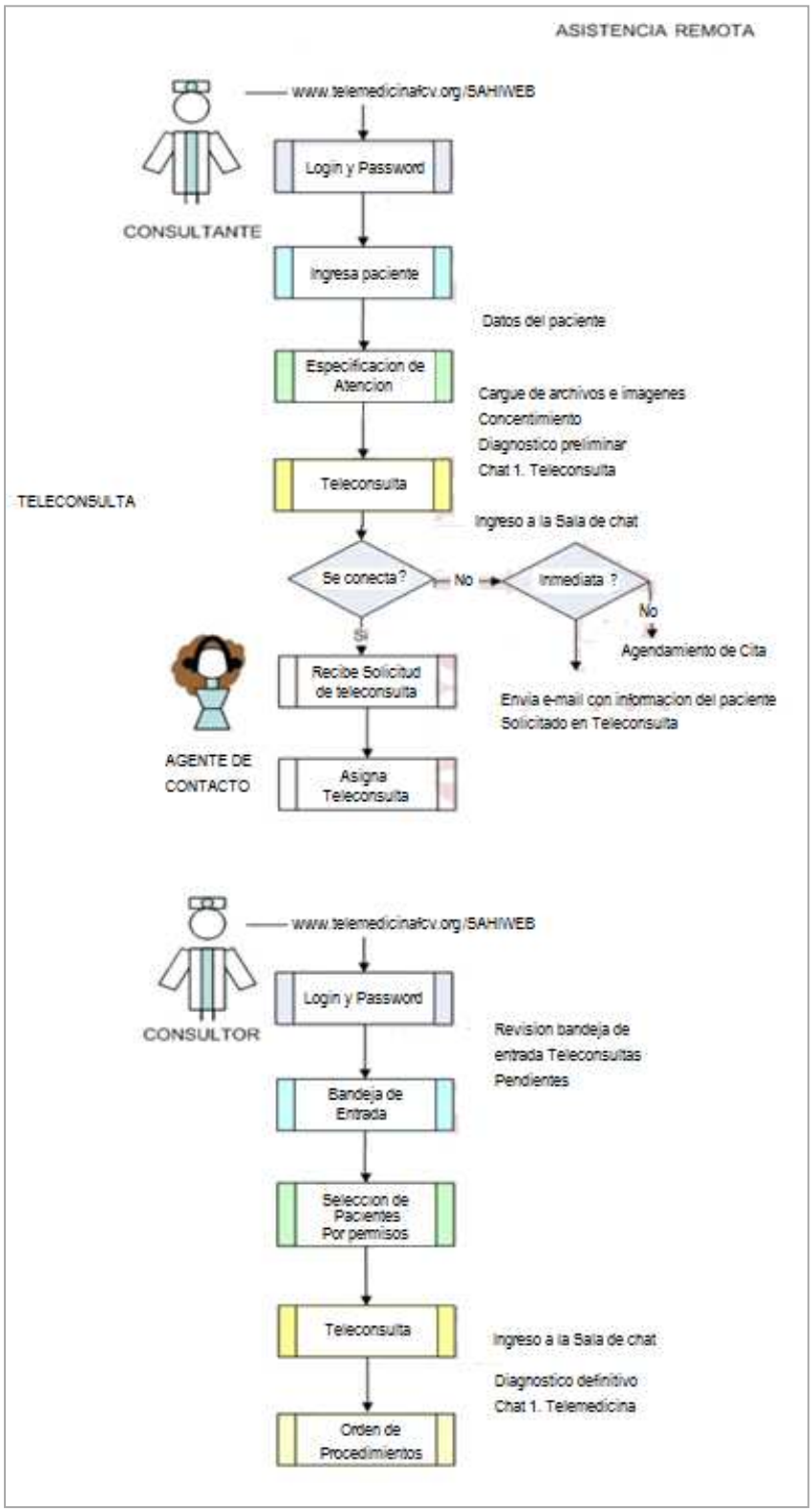

Figura 14. Diagrama de procesos de Tele-Consulta

La gran diferencia que existe comparando Tele-Apoyo Diagnostico, es que la Tele-Consulta se ejecuta en tiempo real, la interacción entre médicos es manipulada por un chat, los doctores expresan sus ideas y sus opiniones en dicho chat.

La conversación y los datos de la solicitud de la Tele-consulta quedan almacenados en una celosa base de datos, obviamente por razones de seguridad y de términos legales, los doctores pueden imprimir informes de dicha conversación, la cual es denominada “atención”.

El médico consultor puede recibir más de una Tele-Consulta en el mismo momento, atendiendo a dos médicos consultantes por así decirlo, en la conversación también podrán subir archivos asociados a la consulta.

Si por algún motivo de red o aplicación, la conversación se “cae”, la atención queda guardad y al igual que a conversación, de tal forma que si se desea retomar la conversación, podrán hacerlo sin ningún inconveniente.

La conversación se dará por terminada, cuando le medico consultante quede satisfecho con el plan de tratamiento o recomendación que le sugiere el especialista, se asigna un diagnostico final. Y se cerrar la conversación.

Para resumir el proceso de Tele-Consulta tenemos:

“El médico consultante de la institución remisoria, debe ingresar al portal http://cnt.telemedicinafcv.org/SAHIWEB/”.

“Diligencia la historia clínica electrónica, y elige la especialidad que desea inter consultar”.

“El Contac Center recibe la solicitud, mantiene la comunicación vía chat con el médico consultante, mientras ubica al Médico especialista de disponibilidad”.

“El médico especialista del centro de referencia analiza la historia clínica, los estudios adjuntos, (Radiografías, electrografías, etc.), interactúa vía chat con el médico consultante, define un diagnostico de impresión, y recomienda la conducta a seguir”.

“El médico consultante define destino, cierra la atención y tiene la opción de imprimir el registro de la atención”.

La característica especial de este proceso, es la respuesta e interacción de los doctores implicados en tiempo real. La corrección y recomendación se hace inmediata, logrando una mejor atención en casos particulares presentados en hospitales donde no hay presencia de doctores especialistas.

3.2.3. Tele-UCI

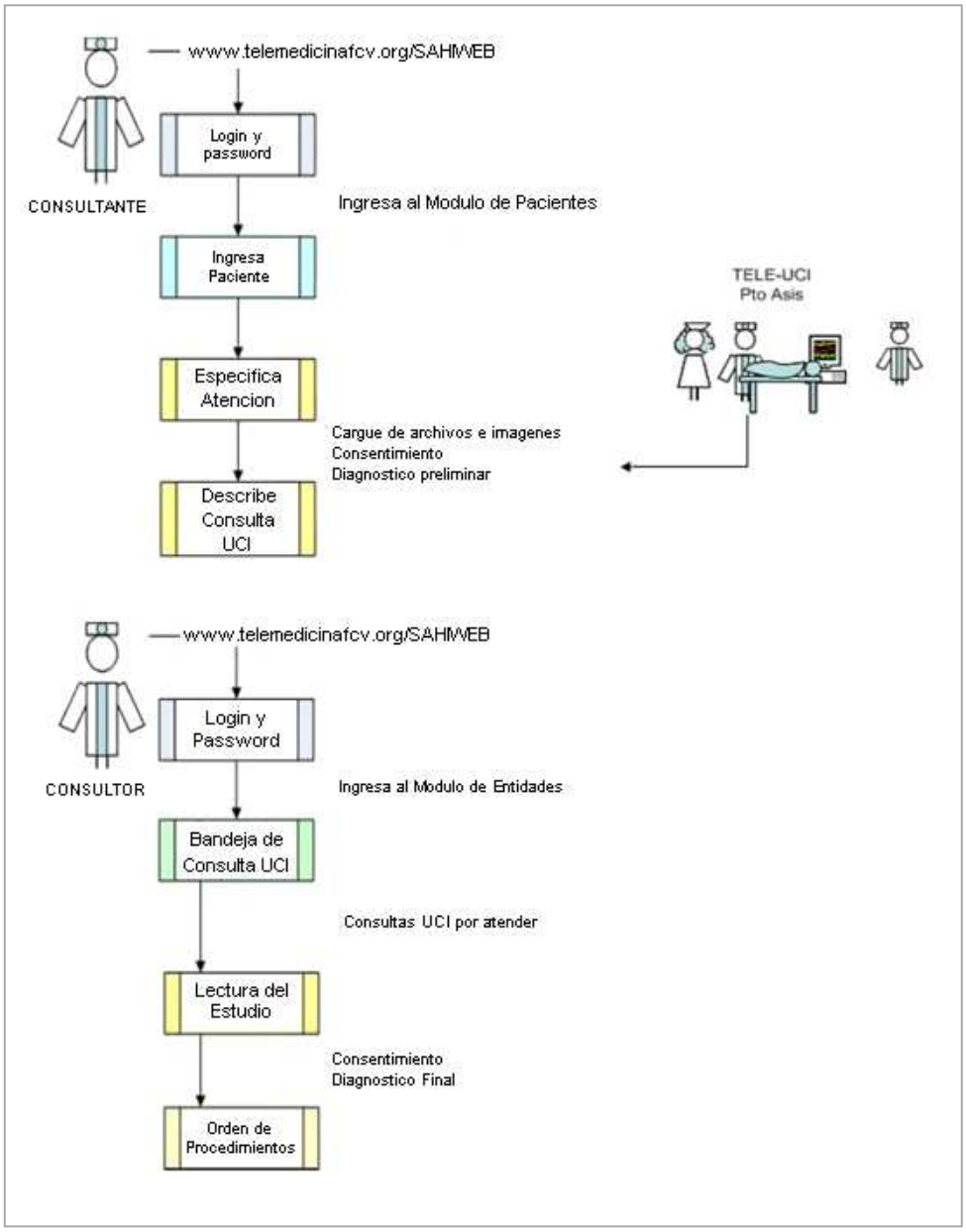

Figura 15. Diagrama de procesos de Tele-UCI

Un grupo de médicos especialistas acceden remotamente a la monitoria del paciente a la monitoria del paciente ubicado en la remisora.

El médico tele experto del centro de referencia tienen acceso en forma remota a la información del monitor de signos vitales y a la historia clínica del paciente, de tal manera que puede, durante las 24 horas, monitorear los parámetros clínicos del paciente e interactuar con el personal de salud de la institución remisora.

En esta sección se lleva todo un seguimiento, a un paciente internado en UCI se le lleva control de ingreso, control de egreso, formulación, evolución.
Este proceso es llevado a cabo por medio de la herramienta cliente –servidor (SAHI).

3.2.4. Monitoreo

Principalmente utilizado en pacientes internados en las unidades de cuidados intensivos de hospitales ubicados en diferentes partes del país, los cuales conectados por medio de equipos producidos por Bioingeniería, transmiten los signos vitales a una oficina de monitoreo, ubicada en la sede principal (Instituto del Corazón Bucaramanga).

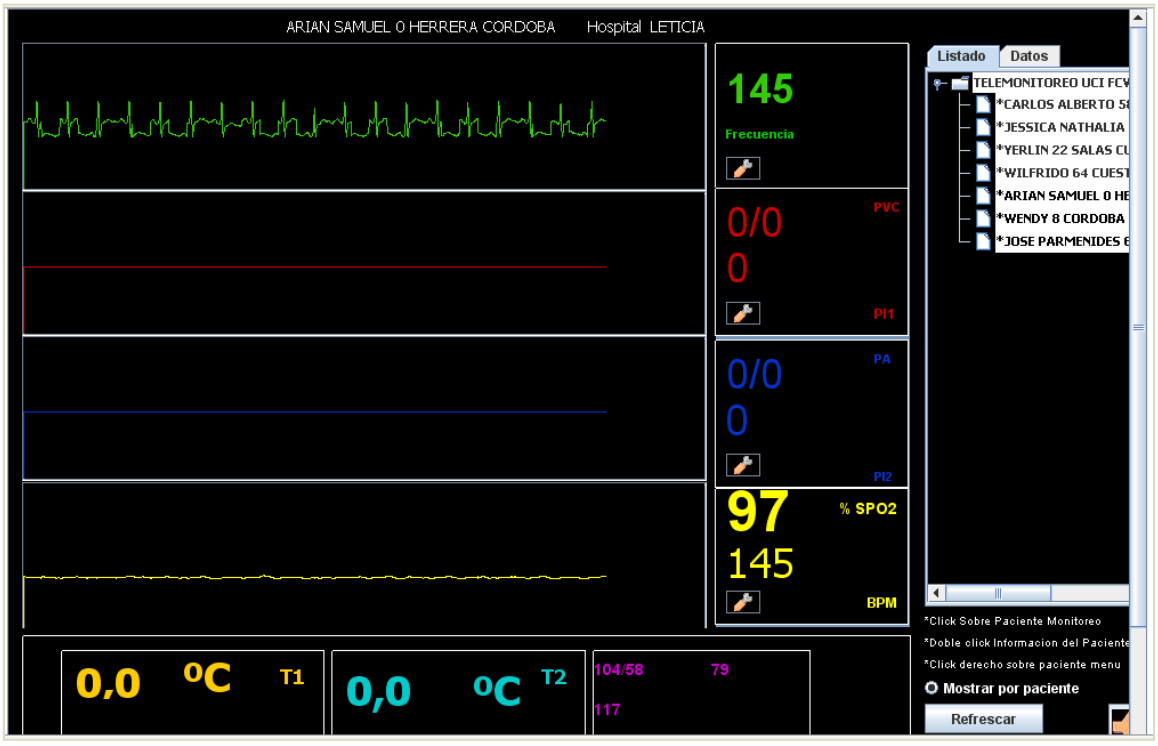

Figura 15. Centro Nacional de Telemedicina - Monitoreo

El objetivo principal del monitoreo, es ofrecer a entidades remisoras la capacidad de brindar un optimo servicio, con calidad de conocimientos médicos locales y conocimientos especializados de doctores de otras partes del país, que brindan soporte las 24 horas del día en conjunto al grupo de trabajo Help Desk.

Logrando de esta forma, una reducción del índice de pacientes fallecidos en las unidades de cuidados intensivos de hospitales que no cuentan con personal especializado dentro de sus instalaciones.

Para destacar, que el servicio de monitoreo ofrece la facilidad de estar pendiente del paciente las 24 horas del día, de manera personal y online,

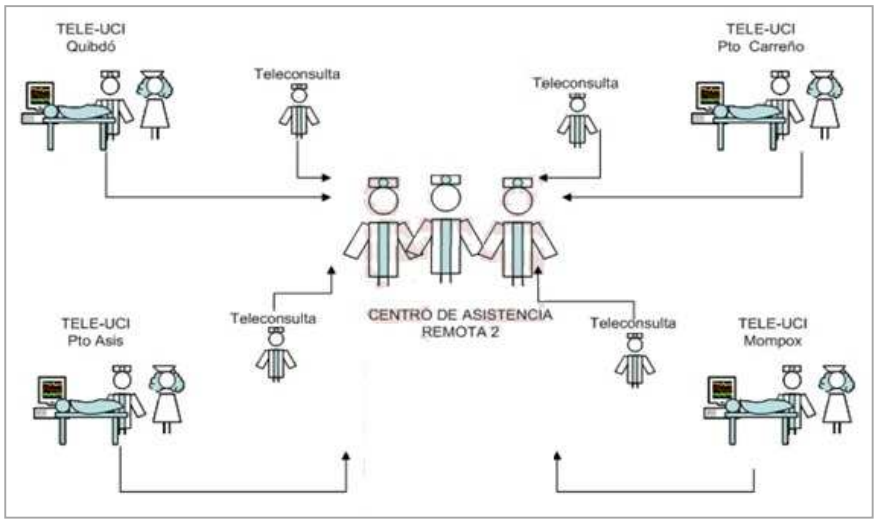

Figura 16. Esquema de monitoreo

El monitoreo actualmente, funciona gracias a una aplicación de escritorio, es decir, que para poder recibir la transmisión de los datos de los signos vitales es necesario tener instalado un software desarrollado por la FCV.

3.2.4.1. Video Paciente

Teniendo en cuenta la descripción anterior de monitoreo, video paciente es un gran complemento, video paciente, transmite por UTP imágenes en tiempo real del paciente internado, ya sea en la unidad de cuidados intensivos o cualquier habitación del hospital que cuente con el servicio de cámaras.

Este Módulo brinda la perspectiva de sitios estratégicos de la habitación del paciente, como lo es el suministro de líquidos, la ubicación de las zondas, etc.…

El paciente es observado las 24 horas del día, gracias a Help Desk, que debe informar cualquier anomalía presentada por el paciente, al hospital a cargo.

Monitoreo y Video Paciente son aplicaciones que actualmente están brindando un servicio óptimo a la comunidad pero de manera disyunta, se tiene programado y proyectado la unificación futura en la aplicación **SAHIWEB**.

3.3. Descripción por Módulos.

3.3.1. Módulo "Login y Password"

Figura 17. Módulo "Login y Password"

El login corresponde a la cedula de ciudadanía del usuario, y el password es el asignado por el administrador del sistema, aquí se realiza una estricta validación y verificación de los datos suministrados, identificando claramente parámetros tales como, nombre del usuario, roles, especialidades, entidad medica a la cual pertenece, hora de ingreso y tiempo de instancia dentro del sistema, entre otros.

Cabe resaltar, por protocolos de seguridad, se desarrollo un método que consiste en que un usuario solo podrá mantener una sesión activa a la vez. Es decir, si ya ingreso al sistema desde algún sitio, nadie más podrá ingresar al sistema con ese usuario en ese momento desde otro sitio.

Se creó un formulario para la validación de ingreso, dicha validación se debía realizar con la tabla “ASI_USUA”, donde los parámetros base para la validación eran la cédula del usuario y clave. En el caso que el usuario tenga acceso a varias entidades se debía mostrar una lista con las entidades.

Se implemento un menú dinámico, donde se debían cargar opciones por usuario, de acuerdo a la configuración contenida en las tablas “ASI_ROLE” y “ASI_MENU_ACCE”.

Desde un inicio, el proyecto siempre estuvo direccionado a ser parametrizado casi en su totalidad, desde base de datos, con lo anterior, todas pantallas, formularios, menús y botones, son configurables desde base de datos. Por lo anterior se debió crear campo en “ASI_MENU” para el re direccionamiento a cada formulario y configurar la asignación de los menús a los usuarios desde “ASI_MENU_ACCE”.

También se implemento un manejo de perfiles para lograr una óptima administración de la aplicación y una confiable y celosa seguridad de la información, de acuerdo al modelo actual de la FCV “SAHI”.

Se desarrollo la aplicación, enfocándose en que el usuario no debe requerir ningún “pluggin” adicional, ni debe necesitar alguna configuración del sistema operativo, para ser funcional. Se hace referencia a lo anterior ya que el portal anterior de Telemedicina, necesitaba los siguientes “pluggins” y requisitos para ser funcional como lo son:

- Puertos Habilitados:
    - 8443 Pagina principal
    - 1500 Chat médicos
    - 1501 Chat Contac center
    - 8080 Dcom
- Desbloquear Ventanas Emergentes
- Flash player
- Visor PDF
- Java v1.5 o superior

Para la nueva aplicación SAHIWEB, no es necesario pre configurar nada de lo descrito anteriormente. Por tal motivo, se hizo énfasis, en que el usuario debe poder ingresar a la aplicación sin tener Flash instalado.

3.3.2. Módulo "Pacientes"

Se diligencian todos los datos personales del paciente, para el llevar el debido seguimiento clínico para su correcta elaboración de historia clínica en web, según la norma establecida a nivel nacional.

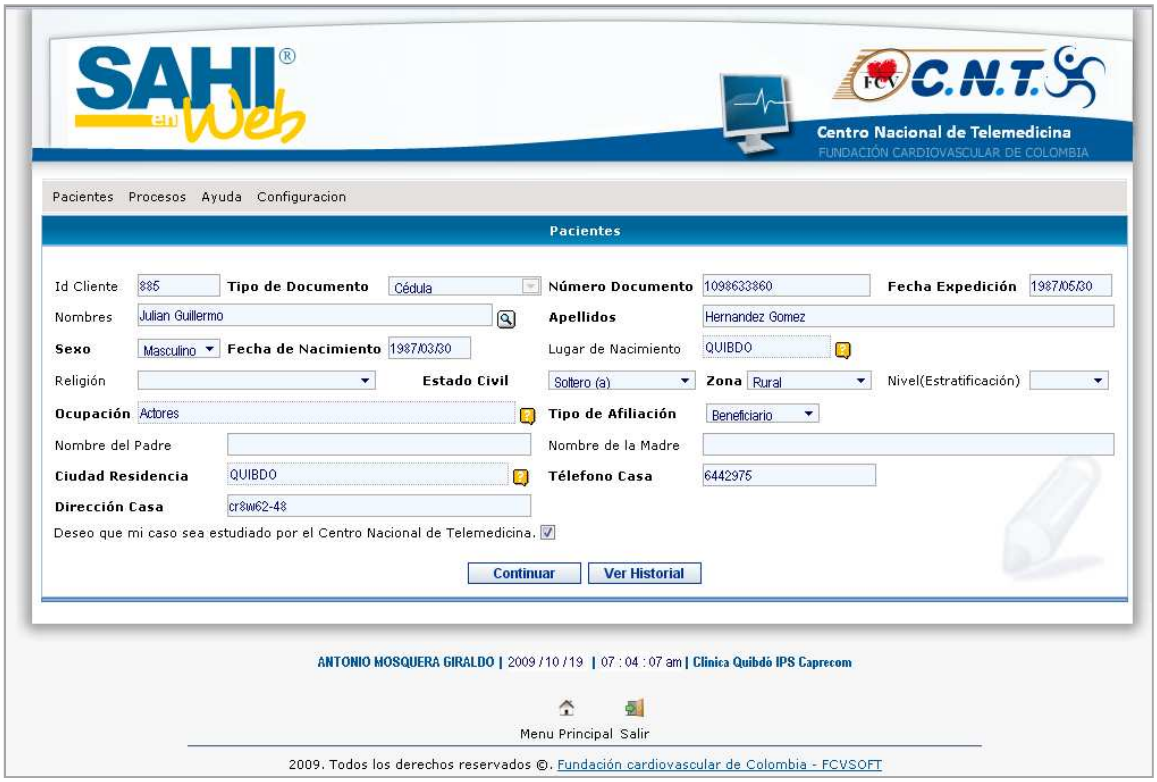

Figura 18. Módulo "Pacientes"

Se validó ardua y exhaustivamente el Módulo de pacientes (donde se crean los pacientes), ya que debe existir un solo paciente en el sistema de información. Tal requerimiento incluyó depuración de registros actuales en la base de datos.

Se crearon los módulos de Pacientes y Atenciones, llenando al cargar los combos desplegables con las respectivas informaciones contenidas en las tablas "genListas" y "HceListas".

Utilizando AJAX (**A**synchronous **J**avaScript **A**nd **X**ML "JavaScript asíncrono y XML", es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir,

en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones) se validó si el paciente es paciente antiguo (ya creado) o nuevo paciente, con solo digitar la cedula, la aplicación verifica la existencia del paciente, si es existente, cargará automáticamente los datos del paciente.

Dado que la historia clínica electrónica de cada paciente debe ser leída con la debida autorización del paciente, se incluyó un check en la pantalla de creación de pacientes, el cual señalarse cuando el usuario está de acuerdo con la metodología de atención del CNT y da su consentimiento para que el caso sea analizado por Telemedicina.

Debido a que el servicio de Telemedicina es brindado a lugares remotos del país, no todos los pacientes cuentan con su debida cedula de ciudadanía o tarjeta de identidad, en el Módulo Pacientes se implemento un método de verificación que se ejecuta cuando se seleccionara "Adulto sin identificación" y "Menor sin identificación" resumiendo el objetivo de este método, es que el usuario no debe poder ingresar ningún valor en la cacilla numero de documento, se le asignará como número de identificación un consecutivo de la tabla "SISTABLA", entre otras cosas.

Teniendo en cuenta que el registro de un paciente debe realizarse los más rápido posible (Ya que en ocasiones el estado del paciente no está en condiciones de llenar un formulario ni mucho menos responder datos personales), la filosofía de los usuarios del sistema era "Entre menos Clicks mejor". Con tal motivo se implemento un método que consistía en, identificar el país, departamento y ciudad de la entidad que registra el servicio, y al carga el Módulo de pacientes, aparezca el país, departamento y ciudad de la entidad, ya que la mayoría de veces, el lugar de nacimiento y lugar de residencia de los pacientes registrados es igual que el de la entidad medica.

3.3.3. Módulo "Atención"

Se diligencian todos los datos correspondientes a la atención o tipo de servicio solicitado, esta pantalla se moldeara dinámicamente dependiendo el servicio solicitado, ya que cada servicio cuenta con un formulario de diligenciamiento diferente.

Aquí debe definir y describir el tipo de estudio solicitado ya se Tele-Uci, Tele-Consulta o Apoyo Diagnostico, aseguradora del paciente, especificaciones y causas de la solicitud de la atención entre otros.

Se Creó un método que registra todos los datos de la atención y los almacena en la tabla “AdmAtencion”, también registra algunos datos de configuración en la tabla “AdmatencionContrato” para lograr una administración y control de los servicios brindados, de acuerdo al hospital que solicita la consulta.

Se ubicó en sitios estratégicos de la aplicación un enlace directo al la información de la historia clínica electrónica del paciente, dicha información contiene un listado de las atenciones que ha tenido el paciente, y si se desea, mostrar información detallada de dicha atención y si es el caso poder imprimir en PDF la información.

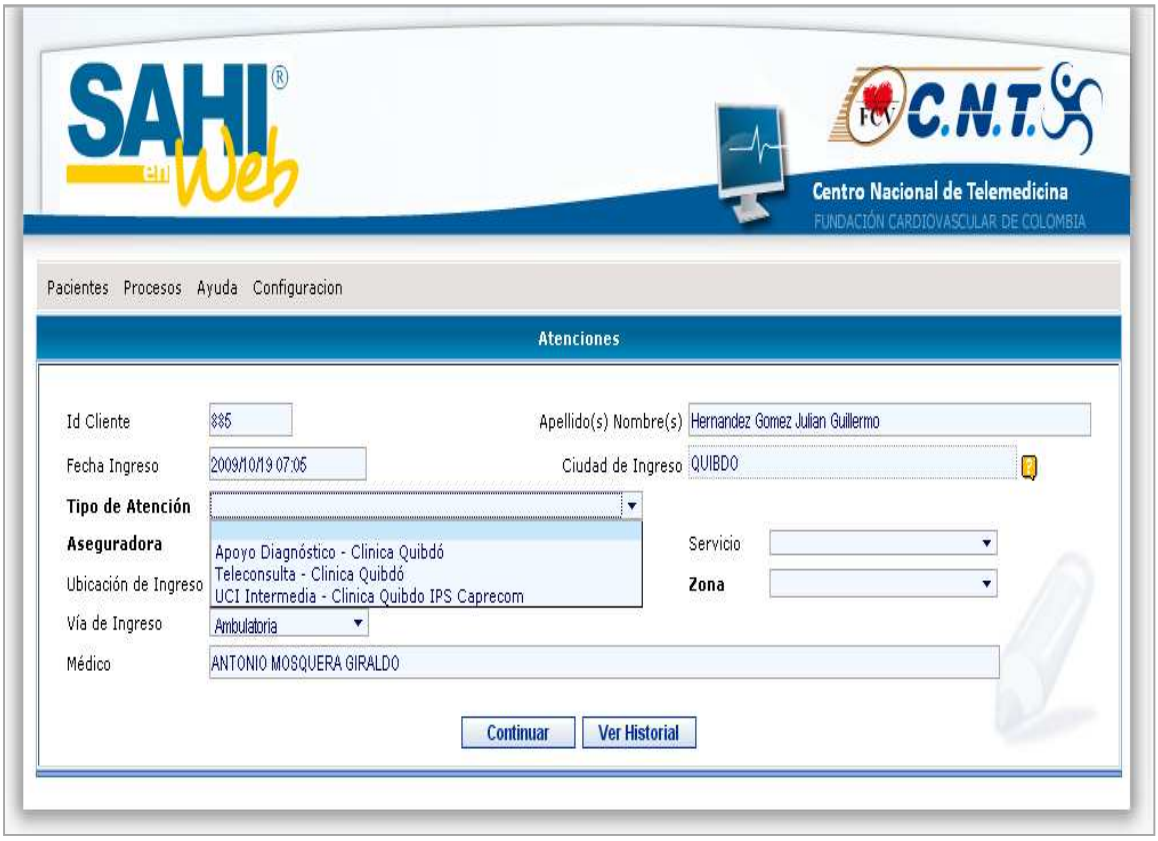

Figura 19. Módulo “Atención”

Se desarrollaron métodos que al seleccionar el tipo de atención la aplicación identifica el esquema de la consulta, cambiando dinámicamente el formulario de datos que se mostrará para diligenciar la información. De igual forma cambia el siguiente Módulo de acceso “Consulta”. Si es Tele-UCI debe enviar al usuario a la consulta con el esquema “Hoja de Ingreso”, si selecciona Tele-consulta debe enviar al usuario a la consulta con el esquema “Tele consulta”, y si se selecciona

“Apoyo Diagnostico” el usuario deberá especificar qué tipo de apoyo diagnostico desea solicitar.

Se implementaros métodos de verificación, que se ejecutaban en el combo de “Tipos de Atención”, este combo siempre debió mostrar todos los servicios brindados, aunque solo se inicio con Tele-UCI.

Se Creó una tabla auxiliar “AdmAtenTipoServ”, para lograr la configurar de los servicios por tipo de atención, con los campos tales como “IdTipoServ, IdAtenTipo, IdServicio, IndHabilitado”, teniendo el modelo de SAHI, se realizaron las respectivas relaciones.

Siguiendo al pie de la letra la norma “1995” de historias clínicas, la cual habla de los campos mínimos requeridos para tener una correcta historia clínica electrónica. Se configuraron los datos obligatorios de todos los formularios, destacándolos en “negrilla”, como se puede observar en la **Figura 19.** los campos Tipo de Servicio, aseguradora y zona son obligatorios.

3.3.4. Módulo “Apoyo Diagnostico”

Se diligencian todos los datos correspondientes para la lectura de las radiografías y los electrocardiogramas, como lo son el código CUPS asociado a las radiografías y la descripción detallada del examen entre otros datos. Este Módulo permite representar las respuestas de dichos exámenes de forma estándar según la exigencia del examen.

También permite diligenciar la descripción del estudio a leer, subir archivos pertinentes a la consulta relacionada, tales como electrocardiogramas y radiografías.

Inicialmente se planeo desarrollar las tres funciones principales en un solo Módulo (consulta), pero en el transcurso del desarrollo se observo que para “apoyo diagnostico” no se necesitaba tanta información, por lo tanto se decidió trasladar este servicio a otro Módulo. Por consiguiente este Módulo fue desarrollado desde cero, teniendo en cuenta el esquema de visualización de los demás módulos.

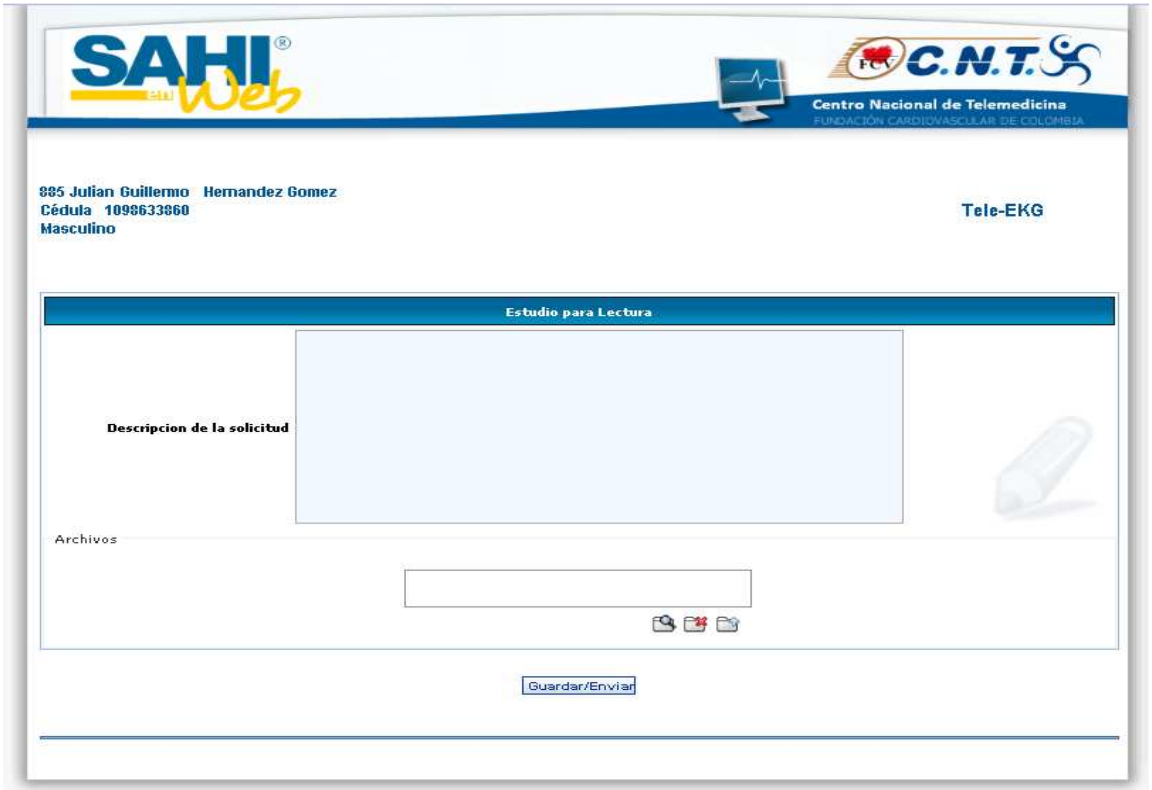

Figura 20. Módulo "Apoyo Diagnostico"

Se desarrollo un método que identifica el esquema de "Apoyo Diagnostico", si el esquema seleccionado es Tele-Radiografía y además se desea asociar archivos con extensión jpg se debe convertir a formato DICOM con los datos del paciente.

También para este esquema se le relaciono un código de procedimiento (CUPS) de la clasificación Imágenes Radiológicas.

Siempre es posible que debido a problemas como caídas de red, bloqueos en bases de datos, entre otros, se diseñó una pantalla que lista los procesos inconclusos, brindando la posibilidad de retomarlos o iniciar uno nuevo, con lo anterior se puede evitar el almacenamiento de información incompleta o redundante.

Figura 20.1 "Pantalla, Listado de Atenciones Abiertas"

Se ajustó el formato de impresión por medio de la herramienta "IReport", para el esquema de Apoyo Diagnóstico, agregándole los datos de solicitud y de respuestas, teniendo en cuenta el código CUPS y el procedimiento seleccionado.

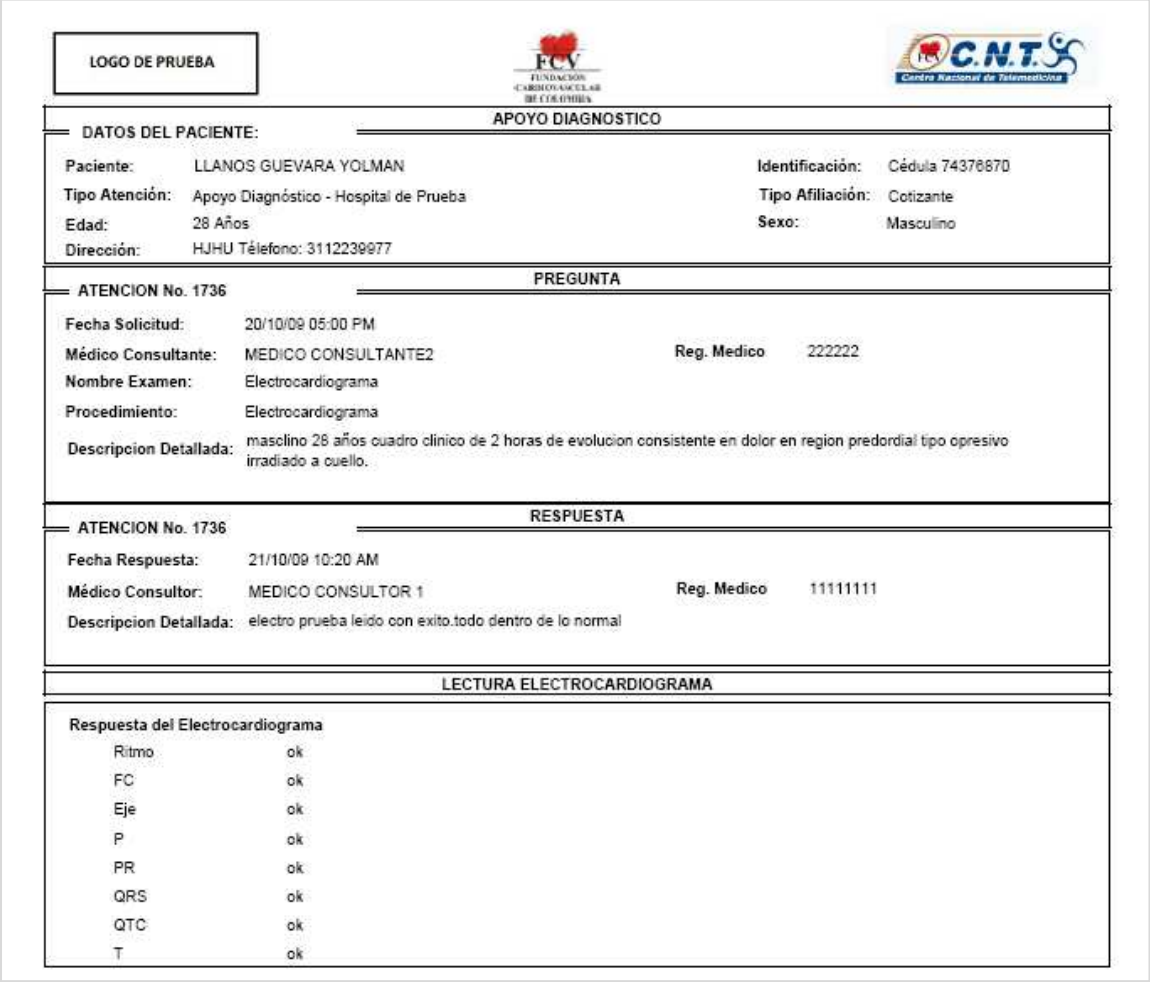

Figura 20.2 "Informe Apoyo Diagnostico"

3.3.5. Módulo “Notas de Enfermería”

Es un módulo creado especialmente para las enfermeras pertenecientes a entidades remisorias, en donde la enfermera puede diligenciar la consulta de la historia clínica electrónica de los pacientes internados en la unidad de cuidados intensivos (UCI), además de manifestar su consentimiento medico, como también las observaciones de dicho paciente.

Se creó una pantalla nueva para notas de enfermería que podrá ser consultada por opción de menú del rol Enfermera Tratante, toda la información diligenciada, se almacena en la tabla “hceNota”.

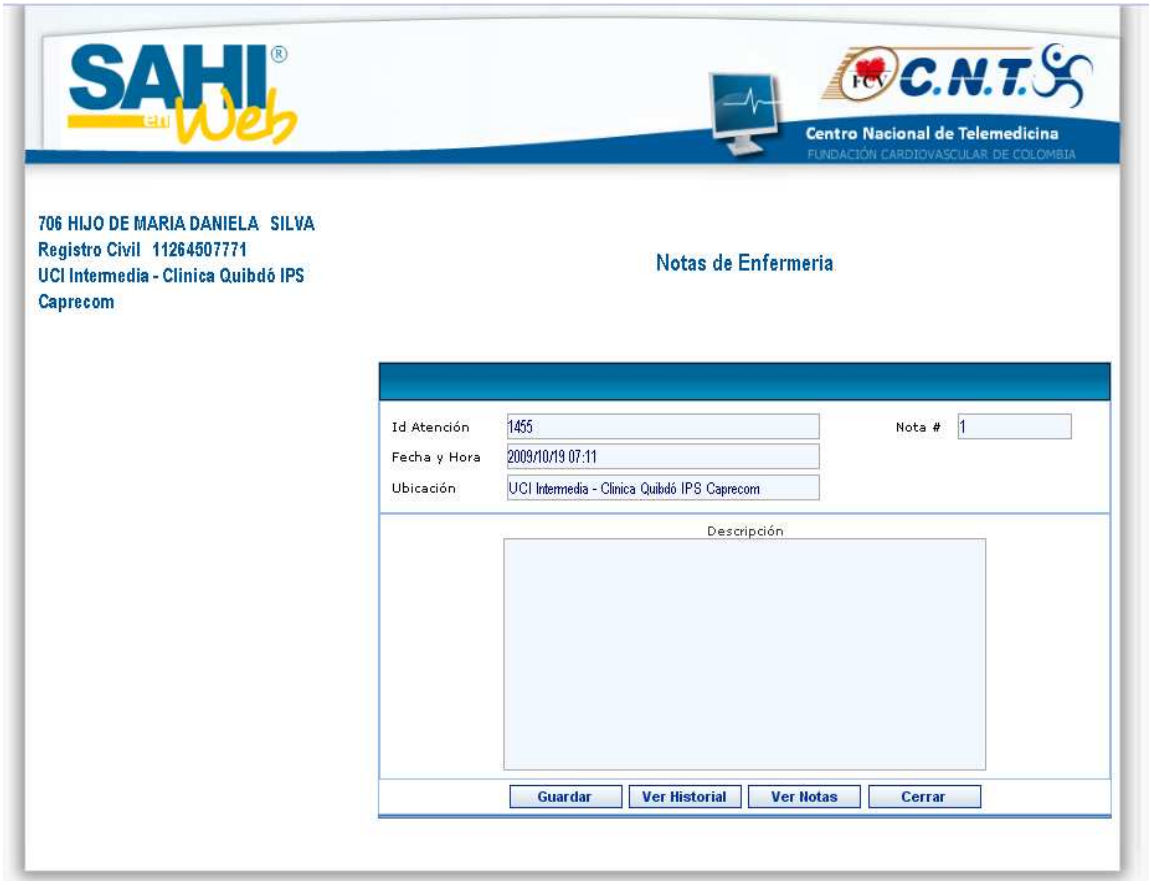

Figura 21. Módulo “Notas de Enfermería”

3.3.6. Módulo "Formulación"

Permite registrar y solicitar una orden de formulación de medicamentos, en este Módulo se sigue un estricto nivel de seguridad en la codificación orientado al despacho seguro y responsable de medicamentos, es decir, cada orden de formulación tiene un único responsable legal, para este caso en el médico consultante recaerá toda la responsabilidad por la consecuencias que se puedan presentar. Así haya recibido asesoría de un medico consultor.

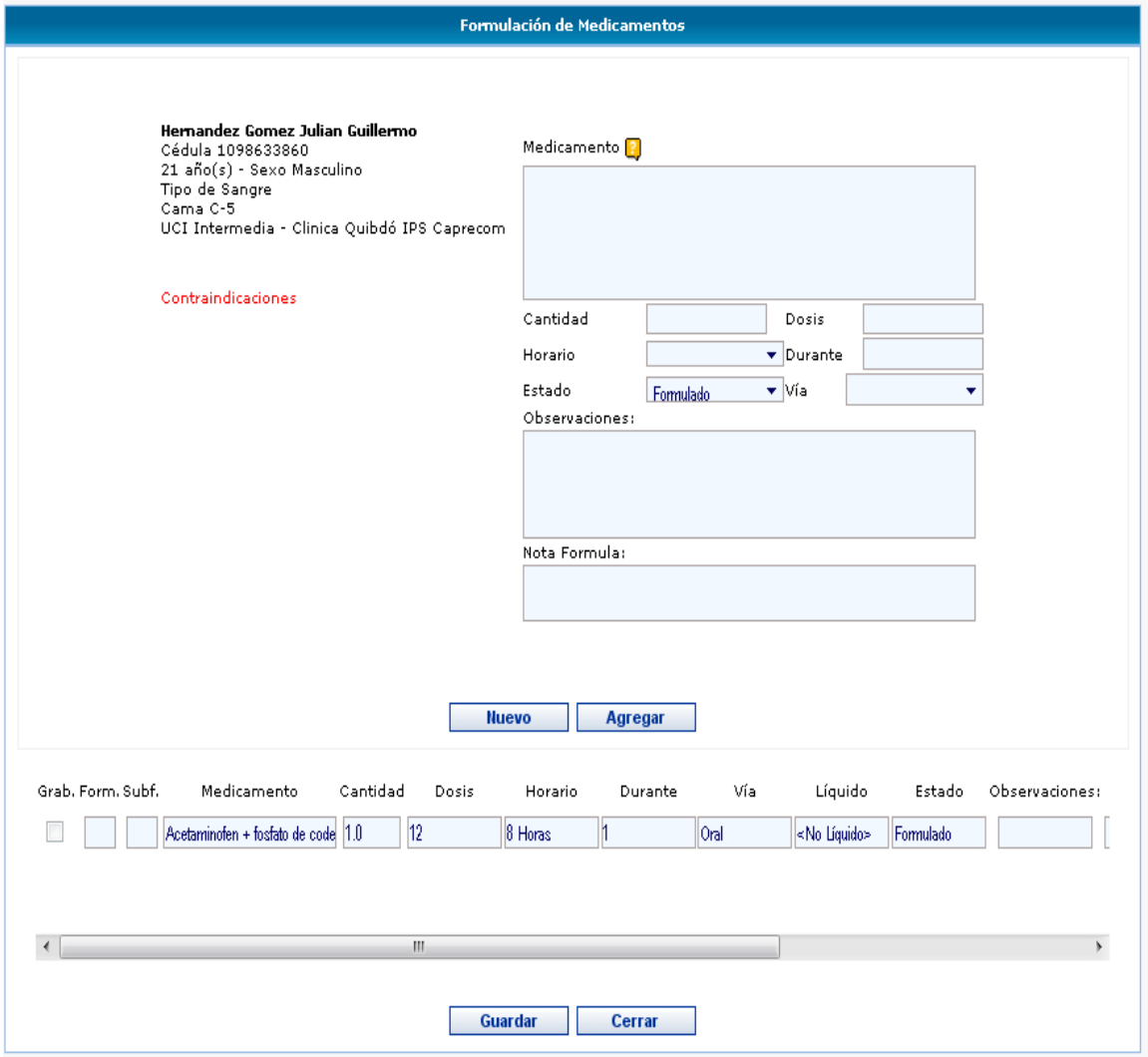

Figura 22. Módulo "Formulación"

En este formulario se realiza la orden de medicamentos, está configurado para mostrarle al usuario si el medicamento ordenado esta en el POS (Plan Obligatorio de Salud del régimen contributivo), modificando a su vez la orden de impresión. Es decir, agregar al nombre del medicamento <NO POS> si es no pos de acuerdo al indicador de almacenamiento en la tabla, esto se debe visualizar en la pantalla y en la impresión.

Se creó un método que carga constantes (logo del hospital y de CNT) de la base de datos con las cuales se realiza la impresión el reporte seleccionado dinámicamente, es decir, el marco de de los reportes puede ser configurado desde la base de datos.

Por criterio de los doctores de telemedicina se decidió Agregar una descripción detallada de los medicamentos formulados, ya que es información fundamental para el paciente.

3.3.7. Módulo "Historia Clínica Electrónica"

Este Módulo es uno de los más importantes de la aplicación ya que contiene toda la información médica de los pacientes, con la cual se llevan a cabo todos los procedimientos médicos.

Este Módulo tiene pendiente un protocolo de seguridad, ya que se debe restringir el acceso a la información de los pacientes, actualmente se espera la aprobación de una norma para la lectura de las historias clínicas, ya que la única persona que puede autorizar la lectura de la historia clínica es el paciente.

Para una acceder de una forma fácil y manejable a la información de la historia clínica de un paciente, se creó un esquema de visualización como se ilustra el la figura 23 "árbol desplegable o árbol de directorios" en donde se imprime en pantalla la siguiente información de la Historia electrónica del Paciente: Datos Ingreso, Resumen de Estancia (Evoluciones, medicamentos, ordenes de procedimiento en orden del mas nuevo al más viejo) y datos de egreso que corresponden a la ultima evolución.

Figura 23. Módulo "Historia Clínica Electrónica"

Teniendo en cuenta que la información de la historia clínica electrónica del paciente puede ser muy extensa, lo cual podría llegar a generar una respuesta lente de la aplicación. Por tal motivo se utilizo nuevamente el servidor AJAX. Ejecutando cada consulta cada vez que se seleccione una rama del árbol.

3.3.8. Módulo "Egreso Paciente"

Se encarga de registrar toda la instancia medica de un paciente en el hospital, aquí se encontrara un resumen descriptivo de los medicamentos formulados y suministrados, un resumen de signos vitales y un resumen de asistencia remota. Es decir, permite el registro de la información de egreso de los pacientes.

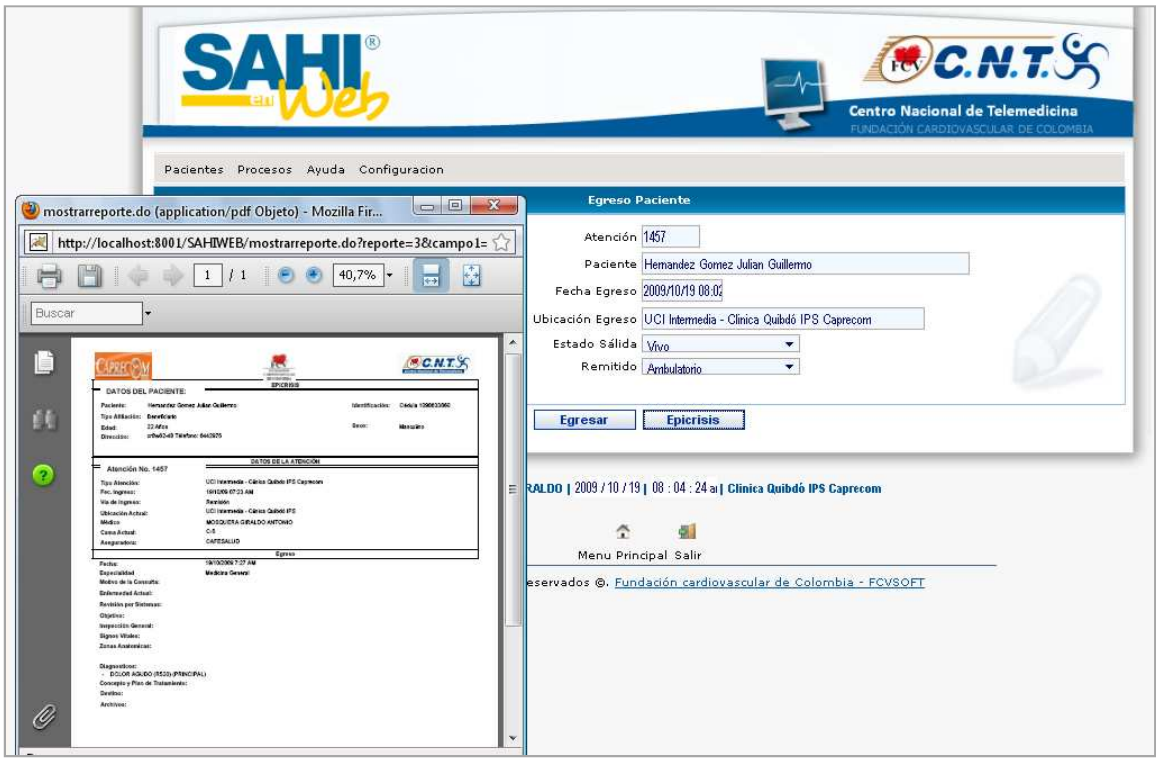

Figura 24. Módulo "Egreso de Pacientes"

Se creó un formulario para dar egreso al paciente ilustrado en la figura 24, en donde se selecciona estado del paciente (vivo o muerto). Automáticamente se libera la cama y se cierran los esquemas que se encuentren abiertos. Para realizar el egreso se debe haber realizado la "Epicrisis" (resumen de la enfermedad que es entregado al paciente cuando éste se da de alta). El botón de Egreso se habilita por opción de menú a los usuarios médicos tratantes y enfermeras.

3.3.9. Módulo "Consulta"

Si el Módulo de Historia Clínica es el corazón de esta aplicación, este Módulo seria la columna vertebral de la aplicación, ya que maneja dos de los tres procesos importantes de la aplicación, que son Tele-UCI y Tele-Consulta, lo cual significa que este Módulo cambiará dependiendo de la solicitud del servicio. También permite diligenciar la descripción del estudio a leer, subir archivos pertinentes a la consulta relacionada, tales como electrocardiogramas y radiografías.

Figura 25.1 Módulo "Consulta (Tele-UCI)"

Se creó un formulario "Consultas" con diseño de acordeón en donde se despliega la información de cada sección o pestaña de forma vertical. Hubo secciones predefinidas (pintadas) que se pueden ocultar y sesiones que se puedan armar de acuerdo a una configuración de la tabla "HceParametro". Los nombres de los títulos de las secciones y de las etiquetas se pueden cambiar de acuerdo a parametrización. Al salir del formulario se creó un método que se ejecuta cerrando el proceso de consulta (cerrando las tablas "hceEsquemasdeAte" y" hceConsultas").

En el formulario de consultas se agrego una pestaña para los “Paraclínicos”, el cual contendrá una caja de texto abierto y la opción de poder subir archivos. Teniendo en cuenta que la pestaña de “Paraclínicos” será visible solo para esquemas tales como: “Hoja de ingreso” y “Manejo intrahospitalario”, recordando que estos servicios pertenecen a tele-UCI.

Debido a que los procesos de consulta dependen de datos anteriores, el orden de configuración de las pestañas fue configurado desde la tabla “HceParámetro”.

Se consulto con el DBA de la Fundación y con la administradora de red el lugar más indicado para el almacenamiento de los archivos del proyecto: hojas de texto, imágenes DICOM, imágenes EKG, entre otros. Se configuro de tal forma que los archivos se almacenaran en dicho lugar.

Dado que las impresiones de ciertas consultas ocupan mucho espacio y contienen demasiada información, se realizo la optimización del espacio, con el fin realizar de impresiones de la consultas en una sola hoja. Esta configuración contiene el logo del hospital que es cargado de una constante desde “GenConstReporte”.

En la pantalla de selección de pacientes por UCI, se desarrollo un método que consiste en realizar una búsqueda por nombre del paciente y cama de la unidad de cuidados intensivos de la entidad, que muestra los pacientes de la ubicación(es) a las que el usuario consultante tenga permisos. Si al seleccionar un paciente dicho paciente no tiene una hoja de ingreso abierta, la aplicación envía al usuario al esquema de consulta “hoja de ingreso” y si ya la contiene debe enviarlo al esquema de “evolución”.

Figura 25.2 Módulo "Consulta (Tele-Consulta)"

3.3.10. Módulo "Cerrar Sesión"

Figura 26 Módulo "Cerrar Sesión"

Este Módulo se desarrollo completamente, al ingresar a la aplicación se mostrara una lista de los usuarios que en ese momento están conectados al sistema, es aquí donde el administrador tiene la opción de desbloquear la cuenta si por algún motivo un usuario abandonó la aplicación de forma incorrecta. Cabe recordar que si el usuario abandona la aplicación sin cerrar sesión, dicho usuario no podrá ingresar a la aplicación desde otro computador, hasta que dicha sesión no se cierre de forma correcta.

1.1.1. Módulo "Actualizar Datos Usuario"

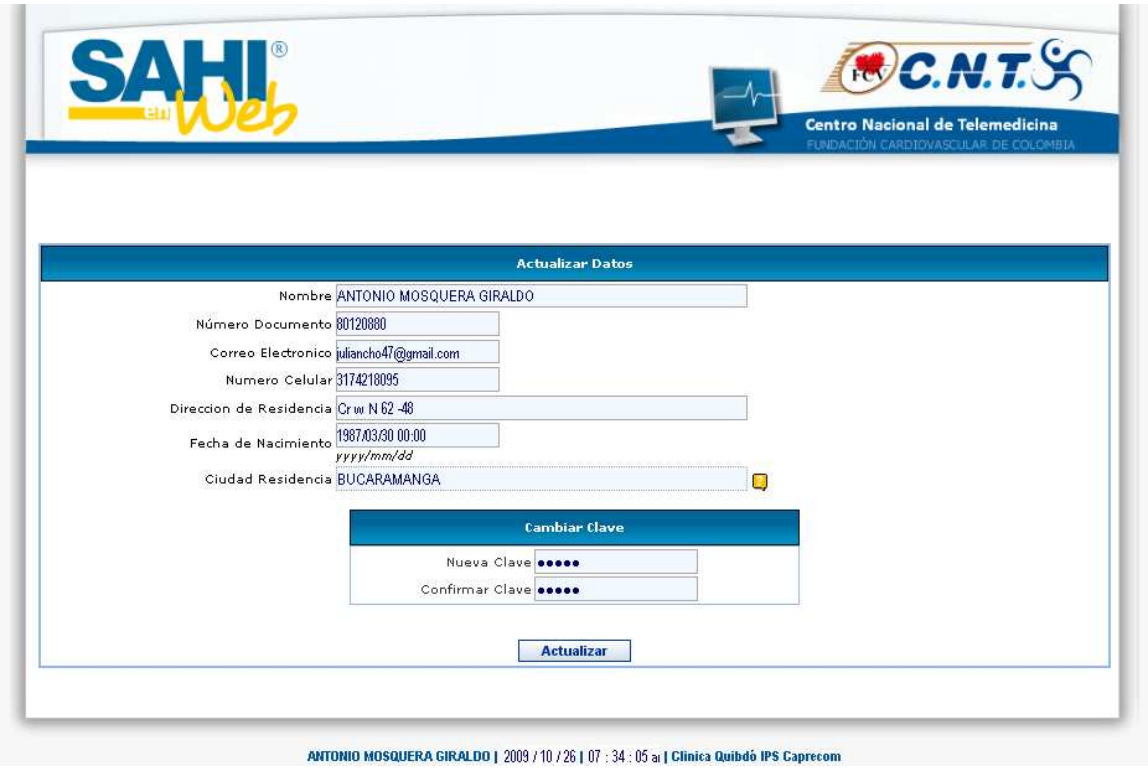

Figura 27 Módulo "Actualizar Datos Usuario"

Este Módulo se creó completamente, el cual tiene como objetivo cumplir con la actualización de datos de los usuarios, este Módulo contiene un formulario que de entrada muestra la información actual del usuario, contiene métodos de validación y de inserción a la base de datos algo diferentes, se hacer referencia al cambio de clave, esta información debió ser desencriptada a la hora de su consulta y encriptado a la hora de su inserción.

1.1.2. Módulo "Bandeja de Entrada"

Figura 28 Módulo "Bandeja de Entrada"

En este Módulo se modificó la pantalla de entrada de usuarios consultores (Entidades), dicha modificación consistió en visualizar de forma amigable, cuantos eventos o servicios pendientes tienen asignados por entidad, es decir número de Tele-consultas asignadas, número de estudios por leer, pacientes en UCI.

3.4 Análisis, Desarrollo y Documentación

3.4.1. Análisis

En el análisis se tiene muy claro que la definición de los requisitos de la aplicación es el fruto del trabajo conjunto de las partes involucradas en desarrollo: los suministradores, los desarrolladores de la aplicación y los clientes.

En el proceso de especificación de requisitos, los requisitos de software se generan en conjunto entre el cliente y el desarrollador, nunca de manera aislada, ya que:

- El cliente no suele entender el proceso de diseño y desarrollo del software como para poderlo redactar.
- El desarrollador, normalmente no suele entender completamente el problema del cliente, debido a que no domina su área de trabajo.

Se realizaron las siguientes actividades para cumplir con la fase de análisis:

- **Definir los requisitos de la aplicación**

Para la realización del análisis de los requisitos fue necesario hacer un proceso iterativo con el fin de crear una definición o especificación preliminar de los requisitos que debe cumplir el software, a partir de la información obtenida mediante las entrevistas realizadas a los doctores de telemedicina.

- **Definir los requisitos de las interfaces**

Interfaces del software con el resto del sistema y con el exterior. No basto con documentar los requisitos que debe cumplir el software, sino que también fue necesario definir las propiedades que se debieron satisfacer para obtener una interacción eficaz con los otros elementos del sistema como el usuario, el hardware u otras aplicaciones software.

- **Priorización de requisitos**

La Integración de los requisitos se desarrollo en un documento de especificación y se les asigno prioridades, de acuerdo a los requerimientos del cliente. En este proceso el usuario tuvo un papel fundamental ya que la asignación de prioridades se realizo en conjunto en con el cliente en función de su importancia o los beneficios que pudo aportar su cumplimiento.

Gracias al trabajo estricto y riguroso del análisis de requerimientos del sistema se obtuvo un informe con las siguientes características:

- ✓ No ambiguo
- ✓ Completo
- ✓ Fácil de verificar

- ✓ Consistente
- ✓ Clasificado por importancia
- ✓ Fácil de modificar
- ✓ Fácil identificación del origen y de las consecuencias de cada requisito
- ✓ Fácil utilización durante la fase de operación y mantenimiento

- **Definición de la estructura de datos.**

Esta etapa define sobre todo la estructura de los datos que se van a manejar dentro del sistema, su control y la seguridad de los mismos dentro de la aplicación, se tenía especial cuidado en el rendimiento, la optimización y escalabilidad de la aplicación, se conoce de antemano la gran cantidad de información a manejar y el constante flujo de datos dentro del sistema.

El principal objetivo, era evitar que por la gran cantidad de datos a manejar, se generen demoras en la carga de la información para el usuario, retrasos en la ejecución de procesos o pérdida de información relevante del sistema, lo que perjudicaría la confiabilidad del software dentro de la organización.

- **Diseño, documentación y construcción de la base de datos del sistema.**

Teniendo en cuenta la definición de la estructura de datos especificada anteriormente se inicia el diseño de la base de datos utilizando el motor de base de datos SQL Server 2005 previamente aprobada por el cliente, además se hace entrega de la administración de la base al DBA (Administrador de Base de Datos) asignado por la dirección técnica del proyecto, quien se encarga de crear los niveles de usuario de la base, definir la viabilidad de cambios solicitados, así como de mantener informado al equipo de desarrollo de todos los cambios que se realicen a la base; además de mantener actualizada la información concerniente con la documentación de la base de datos entre los que se encuentran los diagramas de entidad – relación y estructuras de datos entre otros.

3.4.2 Desarrollo

Para el desarrollo fue necesario establecer un enfoque disciplinado y sistemático para la realización de la aplicación. La metodología de desarrollo (en cascada), influyo directamente en este proceso de construcción. Lo primero que se tuvo en cuenta al momento del desarrollo fue:

- ✓ Dividir el proyecto por etapas
- ✓ Definir claramente que tareas se llevaron a cabo en cada etapa
- ✓ Definir que salidas se producen y cuando se deben producir
- ✓ Definir las restricciones que se aplican en el software
- ✓ Familiarización de las herramientas de desarrollo a usar

Con la metodología usada se lograron cubrir dos necesidades principales:

- **Mejores aplicaciones**

Un mejor proceso de desarrollo que identifico las salidas de cada fase de forma que se pudo planificar y controlar el proyecto. De esta forma la aplicación se desarrollo más rápidamente y con los recursos apropiados.

- **Logar un proceso estándar en la organización**

Esto aporto claros beneficios: una mayor integración entre los módulos de la aplicación y una mayor facilidad en el cambio del personal de desarrollo del producto.

3.4.2.1 Etapas de Desarrollo

3.4.2.1.1 Primera Etapa

En la primera etapa del proyecto una vez pasado por la etapa de transición que es la etapa del análisis, proseguimos a la primera etapa de desarrollo, en esta etapa se empalmo de la mejor manera con el trabajo y planeación que desde un inicio desarrollo “CreandoSoft”.

La empresa “CreandoSoft” cumplió con un listado de 41 (cuarenta y un) requerimientos generados por Telemedicina, en el momento que comenzó la práctica se había generado una lista de 25 requerimientos adicionales que no cubría el contrato pactado con CreandoSoft y fue desde ahí donde empieza el desarrollo.

Como se explico en el análisis se organizo ese listado de requerimientos por orden de prioridades y necesidades, donde primaba el beneficio del cliente.

Liberación del proyecto y puesta en marcha (Solo TELE-UCI)

Una vez finalizado el desarrollo del listado de los requerimientos generados por el cliente y debidamente documentados (Según el documento de calidad R-DIDES-09) se programó la generación de la primera versión de desarrollo.

Seguidamente, se brindo la respectiva capacitación al grupo de empleados que se encargara de probar el software, el grupo encargado de realizar dichas funciones es el grupo de (Help Desk).

Terminada la capacitación y debidamente informada, se liberó una versión de la aplicación para probar.

Pruebas de la aplicación.

Las pruebas servirán para evaluar todos los componentes de la aplicación para determinar si los productos de una fase dada, satisface las condiciones impuestas al comienzo de dicha fase.

En el proceso de pruebas se tuvo muy en cuenta lo siguiente:

- ✓ Cada caso de prueba se definió el resultado de saluda esperado, este resultado esperado fue el que se comparo con el realmente obtenido de la ejecución de la prueba. Las discrepancias entre ambos (errores) se consideraron síntomas de un posible defecto en la aplicación.

- ✓ El programador no probo sus propios programas porque podría desear (consciente o inconscientemente) demostrar que funcionan sin problemas. Esta actitud inadecuada llevaría a realizar pruebas menos rigurosas de lo deseable, en este caso se aseguro realizar una búsqueda implacable de cualquier error cometido.

- ✓ Se inspecciono a conciencia el resultado de cada prueba para, así, poder descubrir posibles síntomas de defectos.

- ✓ Al generar casos de prueba se incluyeron tanto datos de entrada validos y esperados como no validos e inesperados.

- ✓ Las pruebas se centraron en dos objetivos:

- Probar si el software no hace lo que debe.

- Probar si el software hace lo que no debe, es decir si provoca efectos secundarios adversos.

✓ Se evitaron los casos desechables es decir, los no documentados ni los diseñados con cuidado, ya que suele ser necesario probar una y otra vez el software hasta que quede libre de defectos. No documentar o guardar los casos significara repetir constantemente el diseño de casos de prueba.

✓ La experiencia en este proceso nos dice que donde hay un defecto hay otros, es decir la probabilidad de descubrir nuevos defectos en una parte del software es proporcional al número de defectos ya descubiertos.

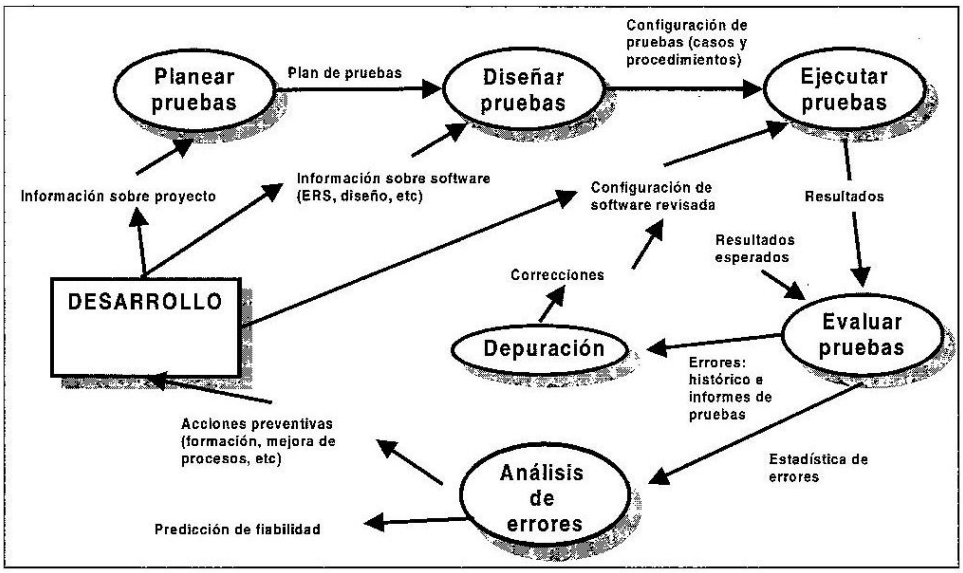

Figura 29. Ciclo completo de las pruebas

Documentación del diseño de las pruebas

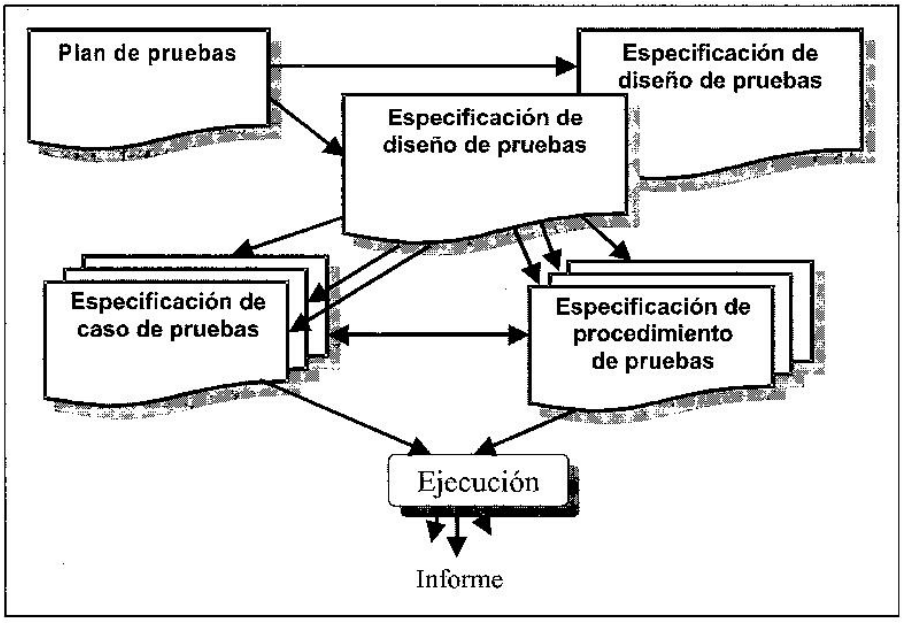

Figura 30. Documentación relacionada con el documento de las pruebas

Las pruebas se documentaron de la siguiente manera:

1. El primer paso se situó en la planificación general del esfuerzo de prueba para cada fase de la estrategia de prueba para el producto.

2. Se genero inicialmente la especificación del diseño de la prueba.

3. Tras generar los casos de pruebas detallados, se especifico como proceder en detalle en la ejecución de dichos casos (procedimientos de prueba).

4. Tanto las especificaciones de casos de prueba como las especificaciones de los procedimientos fueros los documentos básicos para las ejecuciones de las pruebas. No obstante son los procedimientos los que determinaron realmente como se desarrolla la ejecución.

- **Corrección de errores generados en las pruebas.**

Una vez recibido el informe generado por las pruebas se procede a depurar la aplicación, es decir a localizar, analizar, y corregir los defectos que se sospecha que contiene el software, las consecuencias suelen ser dos.

- ✓ Encontrar la causa del error, analizarla y corregirla

- ✓ No encontrar la causa y, por lo tanto, tener que generar nuevos casos de prueba que puedan proporcionar información adicional para su localización.

A la hora de corregir los errores se tuvo muy en cuenta que lo que debe corregirse y hacer desaparecer es el defecto, no sus síntomas. Y la probabilidad de corregir perfectamente un defecto no es del 100%, por consiguiente se debieron revisar las correcciones antes de implantarlas.

Se tomaron las medidas de cuidado necesarias a la hora de corregir, ya en este paso, podrían crearse nuevos defectos.

La corrección se situó temporalmente en la fase de diseño, se mentalizo en que se reinicia el diseño de la sección de código defectuoso y no solo se retoca el código.

Validación y Verificación.

El objetivo de la validación es determinar la corrección del producto final respecto a las necesidades del usuario.

El objetivo de la verificación es demostrar la consistencia, compleción y corrección del software entre las fases del ciclo de desarrollo de un proyecto. Los procedimientos principales para llevar a cabo dichas actividades son las revisiones de software.

3.4.2.1.2. Segunda Etapa

El proceso de desarrollo en esta etapa, se siguieron los paso para implementación de manera análoga como se hizo en la etapa uno, con la diferencia que en esta etapa, el requisito más importante y por ende principal, era la migración de base datos (CNT a SAHIMULTIEMPRESA).

- **Migración de la base de datos CNT a SAHI.**

En el proceso de unificación o migración, cabe recalcar que no solo se van a unificar las aplicaciones como tal, sino que a su vez, se configurará una base de datos para todas las aplicaciones.

La FCV tiene una administración de varias bases de datos, dos bases de datos en especiales CNT y SAHIMULTIEMPRESA, se encargaban del funcionamiento del portal de Telemedicina antiguo (CNT) y la segunda, se encarga de llevar la administración financiera del personal administrativo y a su vez de administrar las historias clínicas electrónicas de los pacientes.

Cabe resaltar que, este proceso fue una de los más arduos de toda la práctica, el resultado de la migración es el secreto del éxito o fracaso del desarrollo de la aplicación según el cronograma.

Sobra decir, imaginando el escenario “Realizar una mala migración y posterior a eso, dar luz verde a una puesta en marcha de desarrollo,” pudieron haber causado efectos catastróficos en el desarrollo, no solo atrasando el plan de actividades sino provocando largos tiempo de las consultas, elaboración de código innecesario y en alguna ocasión erróneo.

Se analizaron las tablas que en definitiva iban a forma parte de la base de datos unificada, revisando el diseño (modelo entidad relación) de datos definido desde un inicio.

Una vez se completo el proceso de migración se realizo una ardua tarea de validación, verificando el correcto funcionamiento de la aplicación con el nuevo modelo de datos.

- **Efectuar procesos de calidad.**

Todo el proceso de migración fue estrictamente vigilado por el DBA (Administrador de Base de datos) de la fundación, asegurando el éxito optimo del proceso a realizar.
Siguiendo el rigurosamente el proceso de documentación estratégicamente planeado, se documento uno a uno todos los cambios que se iban a efectuar en la

nueva base de datos, todos los “scripts” de las tablas, por medio del documento de calidad.

Pruebas de la aplicación.

Se realizaron las respectivas validaciones a cargo del grupo de Help Desk, análogamente como se describió en la etapa numero uno (ver pág. 69).

3.4.2.1.3. Tercera Etapa

- **Realizar entrevistas usuarios finales.**

Se realizaron varias entrevistas a los doctores de telemedicina para conocer el desempeño de la aplicación y escuchar nuevas sugerencias de la misma. Cabe resaltar que se debe tener muy en claro que no bastó con hacer preguntas para obtener toda la información necesaria. Fue muy importante la forma en que se planeo la conversación y la relación que se estableció con la entrevista.

Para lograr lo anterior se hizo hincapié en poseer ciertas cualidades, tales como:

- Imparcial.
- Ponderado.
- Buen oyente.
- Cordial y accesible.
- Paciente.
- Flexible.

Durante la fase de la entrevista, se documento e investigo sobre la organización cliente, analizando los documentos de la empresa disponibles, esta preparación fue esencial para que el resultado de la entrevista fuera positivo y eficaz, hubiese entorpecido el proceso de entrevista si no se contaran con los conocimientos y conceptos básicos de la actividad.

Sin olvidar, que la entrevista se centro en aquellos aspectos del trabajo no son accesibles por otros medios (como la observación y el análisis de documentos).

Se opto por realizar la entrevista con un enfoque “top-down”, comenzando a entrevistar a los doctores especialistas y gerentes (que pueden ofrecer una visión global) y terminando por hablar con los asistentes de enfermería que aportaron pequeños detalles importantes.

A nivel general fue muy importante transmitir el interés y el entusiasmo que significó el estar desarrollando la aplicación,

- **Estudio de viabilidad y factibilidad (sugerencias o requisitos de los clientes).**

Para realizar el estudio de viabilidad de las sugerencias y/o requisitos generados por los clientes, se tuvo muy en claro los siguientes aspectos:

Económico: Se determino si el beneficio obtenido compensa el coste.

Técnico: Se estudio si la funcionalidad, el rendimiento o las restricciones marcadas son realizables.

Legal: Se analizó, si llegado el caso, los requisitos atentan contra alguna ley, reglamento o contra disposiciones legales de contrato.

Operativa: Se determino si se podía implantar de manera efectiva.

Plazos y Calendario: Se estudio, si las fechas impuestas de desarrollo podrían cumplirse.

- **Liberación del proyecto y puesta en marcha (TELE-UCI, Tele-Consulta y Tele-Apoyo Diagnostico).**

En esta altura de la práctica, el proyecto ha alcanzado una madures importante, la aplicación ya se encuentra en condiciones de salir a producción, es decir, ya podría hacerse una entrega formal del producto, como una primera versión SAHIWEB v1.0.0.0.

En este momento la documentación final de toda la aplicación (procesos, scripts, pantallazos, etc.) han pasado por varios procesos de calidad y revisiones por parte del área de Calidad y el área de Help Desk.

Pruebas de la aplicación.

Se realizaron las respectivas validaciones a cargo del grupo de Help Desk, análogamente como se describió en la etapa numero uno (ver pág. 69).

3.5 Proceso de mejoramiento

- **Proceso de mejora**

Para realizar un proceso de mejora de la aplicación debe tener muy pendiente los siguiente: Analizar el listado de requerimientos generados por el cliente desarrollado y debidamente documentados (Según el documento de calidad R-DIDES-09), justo después, brindar la respectiva capacitación al grupo de empleados que se encargara de validar la aplicación (Help Desk), seguido de esto, se liberaba una versión de la aplicación para validación y pruebas de escritorio.

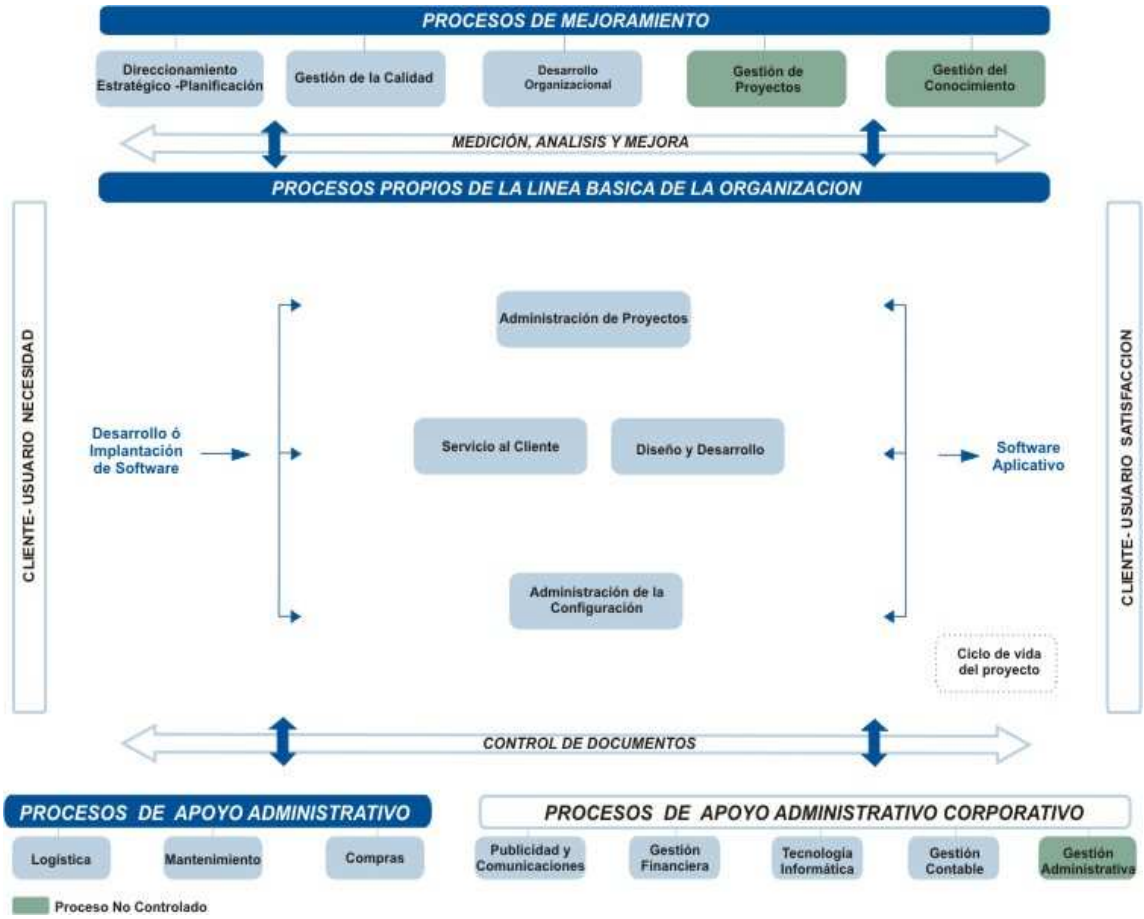

Figura 31. Ciclo "Procesos de Mejoramiento"

Una vez validada la aplicación por el grupo de Help Desk, Realizar las respectivas correcciones de errores presentados o realiza desarrollos recomendados,

después, se prosigue a la elaboración de los manuales de usuario. Y por ultimo brindar la respectiva capacitación a los usuarios finales.

La imagen anterior describe la secuencia de procesos que se debe seguir a la hora de desarrollar una solución informática para un cliente.

4 Conclusiones y Recomendaciones

Conclusiones

- El modelo de práctica empresarial es una gran experiencia para el estudiante que adquiere conocimientos en el área laboral, conoce el funcionamiento de una empresa, y las políticas de calidad implementadas para satisfacer a los clientes.

- La utilización de metodologías avanzadas de desarrollo software, así como el uso de herramientas y lenguajes de diseño unificados (UML), facilitan las labores en las diferentes etapas de realización de un proyecto y reduce el tiempo de ejecución, que implica reducción en los costos y cumplimiento de los cronogramas de actividades.

- La utilización de un framework en el desarrollo de una aplicación implica un cierto coste inicial de aprendizaje, aunque a largo plazo es probable que facilite tanto el desarrollo como el mantenimiento.

- Existen multitud de frameworks orientados a diferentes lenguajes, funcionalidades, etc. Aunque la elección de uno de ellos puede ser una tarea complicada, lo más probable que a largo plazo sólo los mejor definidos (o más utilizados, que no siempre coinciden con los primeros) permanezcan. Y si ninguno de ellos se adapta a las necesidades de desarrollo, siempre es mejor definir uno propio que desarrollar "al por mayor".

- El Framework Hibernate es un marco de trabajo excelente, no solo facilita el manejo de datos dentro de la aplicación, sino que también reduce los tiempos de consulta notablemente.

- SQL Server 2005 es una aplicación excelente para el manejo de datos, pues brinda mucho más que un motor de bases de datos, facilitan el manejo de los datos dentro de una aplicación y garantizan su seguridad.

- Encargarse de actividades que implican el contacto con clientes de la empresa, desarrolla en el estudiante, habilidades en relaciones públicas, muy importantes dentro del medio laboral y que es un área que el ingeniero de sistemas mantiene bastante descuidada

- "Un proyecto software jamás se termina, se abandona", SAHIWEB no es la excepción, el proyecto continuara generando versiones de mejora hasta que de ideas para la realización de un nuevo proyecto.

Recomendaciones

- Es importante que se continúe con la siguiente etapa de unificación de la aplicación actual, para tener un excelente producto software que satisfaga todas las necesidades de los clientes y contribuir al progreso de la empresa.

- Para un mejor avance en el desarrollo del proyecto es conveniente tener un mayor número de programadores en esta fase, teniendo en cuenta la cantidad de tareas que existen actualmente para cada uno de los módulos y por la complejidad de la aplicación SAHIWEB.

- Dedicar un grupo de programadores exclusivamente para el desarrollo del proyecto y otro para el soporte técnico de los productos que actualmente están desarrollados, y con esto poder ofrecer un mejor servicio a los clientes y evitar retrasos en el desarrollo del proyecto.

REFERENCIAS BIBLIOGRÁFICAS

1. Mario G Piattini, José A. Calvo, Joaquín Cervera, Luis Fernández, Análisis y diseño de Aplicaciones Informáticas de Gestión, Colombia: Alfaomega 2004.

2. José Cárcamo Sepúlveda, diseño y Aplicación de Sistemas de Bases de Datos en Entorno WEB con MYSQL y ORACLE, Colombia 2008

3. Jesús Carretero Pérez, Félix García Carballeira, José Manuel Pérez Lobato, José María Pérez Menor, Problemas Resueltos De Programación En Lenguaje Java, España: Agapea 2006.

4. Savit; Wilcox & Jayaraman, Java Para La Empresa, EEUU 2007.

5. Guerrero, Fernando G. Rojas, Carlos Eduardo, Programación en Microsoft SQL Server 2000: con ejemplos EEUU: QUE Publishing 2001.

6. Robinson, Mark, Microsoft Sql Server 2005 Reporting Services For Dummies, EEUU: Agapea 2005

ANEXOS

ANEXO A HOJA DE REQUERIMIENTOS

ANEXO B MODELO ENTIDAD RELACION – SAHIWEB