Universidad de las Ciencias Informáticas

Facultad 7

Trabajo de Diploma para optar por el
Título de Ingeniero en Ciencias Informáticas

Título: Infraestructura de Actualización para el Sistema de Información Hospitalaria alas HIS

Autores: Reinier Reisy Quevedo Batista
Laritza Laguardia López

Tutor: Ing. Alejandro Mario Velázquez Carralero
Co-Tutor: Ing. Reynier Soto Góngora

La Habana, julio de 2011
"Año 53 de la Revolución"

Datos de contacto

Ing. Alejandro Mario Velázquez Carralero

Ingeniero en Ciencias Informáticas, graduado en la UCI, en el curso 2006-07. Posee la Categoría Docente de Profesor Instructor. Se desempeña como Arquitecto Principal de Área Temática Gestión Hospitalaria.

Correo electrónico: amvelazquez@uci.cu

Ing. Reynier Soto Góngora

Ingeniero en Ciencias Informáticas, graduado en la UCI. Se desempeña como Programador del Área Temática Gestión Hospitalaria.

Correo electrónico: rgongora@uci.cu

Resumen

Actualmente en el Sistema de Información Hospitalaria alas HIS se realiza el proceso de actualización de forma manual. Lo que ocasiona gastos innecesarios de recursos humanos y materiales al no contemplar las dependencias con respecto a la base de datos que pueden tener cada uno de los paquetes de actualización. Además, de no tener en cuenta los registros de versiones para cada uno de estos paquetes, lo que puede afectar directamente los procesos que se desarrollan en una institución hospitalaria. Por lo antes expuesto se propone el desarrollo de una Infraestructura de Actualización que garantice la actualización automatizada del Sistema de Información Hospitalaria alas HIS.

El desarrollo de la solución propuesta fue guiado por el Proceso Unificado de Desarrollo RUP, se basa en tecnologías libres, multiplataforma y cuenta con una arquitectura orientada a servicios como: Middleware y la Cliente Servidor. Se utiliza Java como lenguaje de programación, PostgreSQL 9.0 como Sistema de Gestión de Base de Datos, como servidor de aplicaciones JBoss AS 4.2. Como parte principal de las tecnologías de intercambios a utilizar se implementó un servicio web y se utilizó el protocolo de transferencia segura de archivos SFTP. Como herramientas de desarrollo se empleó el Visual Paradigm para UML 6.4 y los IDE de desarrollo: Eclipse 3.4 y Netbeans 6.9.

Entre los beneficios se destaca evitar el retraso las actividades hospitalarias, brindar más tiempo de vida útil al sistema y corregir errores que no fueron detectados antes de liberar el sistema.

Palabras clave: actualización, automática, infraestructura.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Desde los comienzos del desarrollo de la industria de software, su premisa ha sido contribuir en el perfeccionamiento de las tareas que el hombre realiza día a día, lo que ha ayudado a mejorar su calidad de vida. Sectores como la economía, la educación y la salud han sido testigos del vertiginoso avance que el desarrollo de software les ha proporcionado. El área de la salud ha recibido una especial atención por el alto número de procesos que en ella se desarrollan y por su importancia social. La gran cantidad de productos orientados a esta esfera ha generado un aumento de la competitividad y optimización, convirtiéndolos en atractivas ofertas.

La actualización automatizada se ha transformado en un indicador para medir el nivel de eficiencia de un software. Pues actualmente es una vía para mejorar funcionalidades, eliminar secciones obsoletas y corregir errores de un sistema, lo que garantiza a las soluciones de algunas empresas permanecer en uso por más tiempo.

Un Sistema de Información Hospitalaria (más conocido por sus siglas en inglés HIS) es un ejemplo de los beneficios que reporta el desarrollo del software a la atención médica. Su objetivo principal es promover la eficiencia del trabajo humano y de los servicios que se brindan en una institución hospitalaria. Alas HIS es un sistema integral para la gestión hospitalaria, centrado en la historia clínica electrónica única, donde se incluye toda la documentación, imágenes e información que se genere en torno a un paciente. Su principal función es apoyar las actividades en los niveles operativos, tácticos y estratégicos dentro de una entidad hospitalaria, para lo cual hace uso de las TIC[1] para recabar, almacenar, procesar y comunicar información clínica y administrativa. (1).

Actualmente el Sistema de Información Hospitalaria alas HIS cuenta con un procedimiento manual de actualización, el cual será descrito por etapas para su mejor comprensión:

[1]Tecnologías de la Información y las Comunicaciones.

Primera etapa:

En esta etapa el jefe de módulo crea el compactado con los archivos de la actualización referentes al módulo correspondiente, para ello utiliza una herramienta llamada ModuleBuilderAndDeployer (MBD) la cual brinda la posibilidad de seleccionar qué archivos incluir en el paquete de actualización, además de un directorio para salvar el compactado resultante.

Segunda etapa:

En esta etapa el jefe de módulo envía mediante correo electrónico o copia hacia un directorio indicado el paquete de actualización que será recibido por el encargado de desplegar la actualización en el servidor de aplicaciones.

Tercera etapa:

En esta etapa el encargado de realizar la actualización utiliza la aplicación MBD para cargar desde un directorio el paquete de actualización y desplegarlo hacia el directorio definido del servidor de aplicaciones. Una vez copiados todos los archivos se procede a reiniciar el servidor para que los cambios surtan efecto.

Por lo antes planteado, se puede observar que la actualización del sistema se hace con ayuda de una herramienta que facilita parte del trabajo. Aunque la situación es propensa a la introducción de errores humanos en varias partes del proceso. Por otra parte no brinda la información necesaria que permita una actualización o despliegue de forma eficiente. A continuación se hará un desglose por etapas de los principales inconvenientes de la actualización manual:

Primera etapa:

- ✓ En caso de la adición de un nuevo módulo al Sistema alas HIS se necesitaría recompilar la aplicación MBD para que sea capaz de confeccionar el paquete para dicho módulo.
- ✓ El diseño de la herramienta MBD impide que se puedan confeccionar paquetes de actualización para proyectos que no tengan una arquitectura similar al Sistema alas HIS, lo que puede causar problemas si la misma es modificada.

Segunda etapa:

- ✓ Actualmente no se tienen definidas las acciones a tomar en caso que puedan existir problemas relacionados con la red. Dichos errores pueden impedir que el paquete de actualización llegue correctamente al personal encargado.

Tercera etapa:

- ✓ La aplicación MBD no considera los errores que pueden derivarse de la copia de los archivos de la actualización hacia el directorio definido en el servidor de aplicaciones, trayendo consigo que si algún archivo no llega a su destino o no lo hace correctamente el personal encargado no recibiría notificación acerca de este suceso.
- ✓ Se hace necesario reiniciar de forma manual el servidor de aplicaciones, lo cual limita el servicio de la aplicación durante el tiempo que el servidor tome para esta tarea.
- ✓ Si algunos de los archivos que componen el paquete de actualización contiene errores o por alguna razón no es correcto, la aplicación no se desplegará correctamente por lo que se tendrá que recuperar los antiguos archivos de forma manual con vista a ponerla en funcionamiento nuevamente.
- ✓ Se deja en manos del personal encargado la realización de una copia de seguridad a la aplicación desplegada, por lo que puede ocurrir una pérdida de información que interrumpiría el servicio de la aplicación.

De forma general, esta vía de actualización deja de ser factible también por otras razones tales como:

- ✓ Aumenta el tiempo destinado a realizar la actualización de la aplicación.
- ✓ Su uso en un despliegue a gran escala no sería conveniente debido a que:
    - ➢ La conexión vía TCP/IP que utiliza para conectarse al servidor pudiese no estar disponible por razones de seguridad.

- En el caso de que hubiera que actualizar un centro de datos, cada uno de los servidores de tendría que actualizarse de manera independiente, lo que traería como consecuencia un retraso significativo.

- ✓ No contempla las dependencias con respecto a la base de datos que pueden tener cada uno de los paquetes de actualización, por lo que este trabajo se realizaría de forma manual lo que incrementa aún más la introducción de errores humanos.
- ✓ No contempla registros de versiones para cada uno de los paquetes de actualización lo que puede traer confusión y pérdida de información.
- ✓ Las fallas que traería una actualización incorrecta afectarían directamente los procesos que se desarrollan en una institución hospitalaria, lo cual traería como consecuencia la negación de servicios que pueden ser de vital importancia tales como:
  - Atención al paciente.
  - Control de los recursos materiales.
  - Planificación de las actividades médicas.

Por tanto se determina como **Problema a resolver**: ¿Cómo automatizar el proceso de actualización en el Sistema de Información Hospitalaria alas HIS? Enmarcado en el **Objeto de estudio** el proceso de actualización en los software. Mientras que el **Campo de acción** se enmarca en el proceso de actualización en el Sistema de Información Hospitalaria alas HIS. Para resolver el problema identificado se propone el siguiente **Objetivo general**: Desarrollar una Infraestructura que automatice la actualización del Sistema de Información Hospitalaria alas HIS.

Para dar cumplimiento al objetivo general planteado anteriormente se definen las siguientes **tareas de la investigación**:

1. Proponer las metodologías, plataformas, tecnologías, librerías y herramientas que conformarán la solución del problema.
2. Proponer una arquitectura acorde con la solución del problema.
3. Proponer un diseño que complemente la solución del problema.

4. Obtener a través del Proceso Unificado de Desarrollo, los artefactos correspondientes a los flujos de trabajo de “Implementación” y “Pruebas”.
5. Implementar el sistema informático aplicando las pautas de diseño y siguiendo lo establecido en la Especificación de Requisitos de Software.

El desarrollo de la Infraestructura de Actualización para el Sistema de Información Hospitalaria alas HIS reportará beneficios como:

- ✓ Extiende el tiempo de vida útil del sistema, por lo que se podrá prescindir de la inversión en un nuevo sistema.
- ✓ Evita el retraso de numerosas actividades que puedan ser ejecutadas por el equipo de desarrollo, ya que agiliza el proceso de actualización y corrige errores.
- ✓ Contribuye a que el sistema alas HIS continúe el flujo normal de los servicios hospitalarios brindados, debido a que estos no se verán afectados durante el proceso de actualización automatizada.

El presente trabajo se encuentra organizado en 4 capítulos:

Capítulo 1. Fundamentación teórica: En él se analizan algunos sistemas informatizados similares al que se desea desarrollar con el propósito de lograr un mejor entendimiento del problema a resolver, además se muestran las tecnologías y software que serán utilizados.

Capítulo 2. Características de la arquitectura: Se describen los requisitos no funcionales, la arquitectura que será utilizada y se fundamenta. La seguridad es expuesta y se presenta una vista de despliegue.

Capítulo 3. Descripción y análisis de la solución propuesta se describen además las clases más importantes del diseño y se realiza una breve valoración de las técnicas de validación propuestas

Capítulo 4. Validación de la solución propuesta. Se define como método de prueba a utilizar la caja negra y se diseñan los casos de pruebas correspondientes a los Casos de Usos del Sistema.

Capítulo 1: FUNDAMENTACIÓN TEÓRICA

El presente capítulo tiene como objetivos fundamentales profundizar en la situación problémica y fundamentar los principales conceptos. Se analizan los sistemas informatizados existentes relacionados con la problemática planteada, que permitan realizar un mejor diseño de la solución propuesta. Además se realiza un estudio de las tendencias, tecnologías actuales, metodologías y herramientas propuestas para la construcción de la solución.

1.1. Mecanismo de actualización automatizada

El software está constantemente es sometido a un proceso de evolución. Esta evolución está conformada por un conjunto de tareas involucradas en el desarrollo, explotación y mantenimiento del mismo; el cual abarca toda la vida del sistema desde la definición de los requisitos hasta el fin de su uso. Como parte del proceso de desarrollo del software se encuentra el despliegue donde usualmente se garantiza la entrega, instalación y mantenimiento del sistema. Según el estudio realizado: “A Characterization Framework for Software Deployment Technologies” (2), en el despliegue se realizan una serie de actividades: liberación, retiro, instalación, desinstalación, actualización, adaptación, activación y desactivación. ***Ver figura 1***

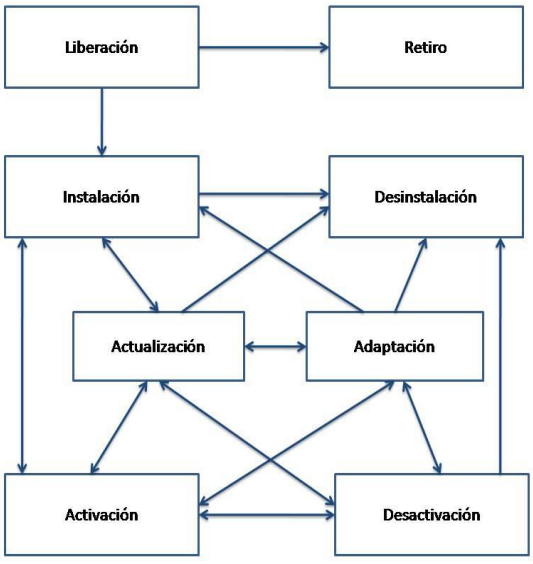

***Figura 1*- Ciclo de procesos desarrollados en el despliegue.**

La actualización es uno de los procesos que se desarrolla dentro del ciclo de vida del despliegue. Puede considerarse como un caso especial de la instalación; aunque es menos complicado porque la mayoría de los recursos necesitados para concebir la actualización ya fueron obtenidos durante este primer proceso. Típicamente el ciclo de vida del despliegue incluye una iteración donde un sistema es desactivado, una nueva versión es instalada y el sistema es reactivado, las cuales conforman las tareas de una actualización.

Para algunos sistemas la desactivación no es necesaria y la actualización pude ser realizada mientras una versión previa esté activa todavía. Similar a la instalación, la actualización incluye la transferencia y configuración de los componentes necesitados para completar sus operaciones. La actualización es generalmente un proceso realizado por los proveedores para garantizar que los clientes mantengan el uso de sus productos.

El proceso de actualización ha constituido objeto de disertación de varios investigadores como: Slinger Jansen, Gerco Ballintijn y Sjaak Brinkkemper el estudio: “A Process Model and Typology for Software Product Updaters” (3), es una muestra de estas investigaciones, es allí donde proponen un modelo general de actualización adaptable a cualquier software. ***Ver figura 2***

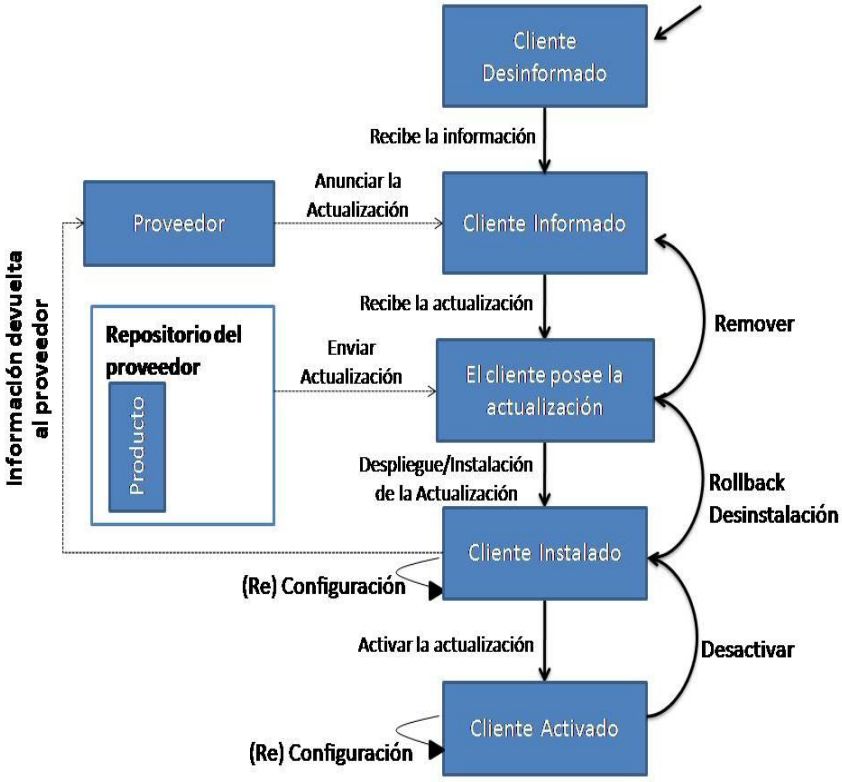

***Figura 2*- Modelo General de Actualización.**

En el modelo general de actualización se describen dos participantes fundamentales: el cliente y el proveedor. El proveedor será la persona o empresa que abastece todo lo necesario para un fin, a grandes grupos, asociaciones y comunidades. (4) En términos informáticos el proveedor será el encargado de la liberación del sistema aunque en algunos casos específicos puede encargarse del retiro de la solución. El proveedor se mantiene al tanto de los procesos que se realizan del lado del cliente, actividades que generalmente forman parte de la implantación de un software.

El cliente sería la persona, empresa u organización que utiliza con asiduidad los servicios de un profesional, empresa o proveedor. (5) También puede ser un programa o dispositivo que solicita determinados servicios a un servidor de actualizaciones. El cliente en este caso será donde se efectúen todos los cambios pertinentes para llevar a cabo con éxito el proceso de la actualización, su función es la de consumir el resultado del servicio provisto por un proveedor.

En el modelo anterior se representa la transición de estados por los cuales el cliente transita, así como la interacción entre el cliente y el proveedor del software. Los bloques representan estados en el diagrama, los estados finales son representados con los bordes subrayados. Las flechas sólidas son transiciones de estados, que se pueden activar tanto por el proveedor como por el cliente. Las flechas discontinuas muestran la interacción entre ambos participantes. Una vez que el proveedor anuncie al cliente que la actualización del software está disponible, comienza el proceso de actualización de acuerdo con los autores de dicho modelo.

Para realizar el proceso de actualización desde un proveedor hacia sus clientes se han creado aplicaciones de software que cumplen este propósito, las cuales son conocidas en la actualidad como aplicaciones actualizadoras o actualizadores; cuyo objetivo fundamental es el de realizar con éxito una actualización, para lo cual cumplen con los requisitos necesarios y atienden las diferentes características del sistema a actualizar. Su acción consiste en trasladar de las manos de los proveedores a las manos de los clientes la actualización de forma eficiente.

Hasta el momento se han identificado cuatro grupos fundamentales de actualizadores los cuales se diferencian por sus procesos de entrega, despliegue así como las políticas y alcance del proceso:

- Herramientas de Despliegue por Paquetes (HDP): Estas tecnologías de despliegue se basan en el concepto de paquetes y en un sitio de depósito o repositorio que almacena la información que representa el estado de cada paquete instalado. Un paquete es un archivo que contiene los ficheros que constituyen el sistema junto a algún metadato que describe el mismo. (3)

- Productos Actualizadores Genéricos (PAG): Estos actualizadores se abstraen completamente del producto a actualizar e intentan ser aplicables al proceso de actualización de cualquier producto. (3)

- Productos Actualizadores del Proveedor (PAP): Este tipo de actualizadores facilitan el proceso de actualización de un solo producto. (3)

- Productos Actualizadores en Caliente (PAC): El propósito de estos es actualizar sin interrumpir el servicio, es decir que la ejecución del programa o sistema no se vea afectada; esto es de gran

utilidad para aplicaciones de tipo críticas como transacciones financieras, sistemas de control aéreo y conmutadores telefónicos. (3)

Para automatizar el proceso de actualización del sistema alas HIS se propone implementar un actualizador del tipo actualizadores del proveedor, el cual ofrece un servicio muy eficiente dirigido hacia la comprobación de la licencia del producto y la interacción con el usuario; este sistema de actualización hará uso del despliegue por paquetes y no interrumpirá el funcionamiento de la aplicación durante el servicio de actualización. Otras actividades como la transferencia de los paquetes de actualización y el restablecimiento de la versión anterior también cuentan con un procedimiento eficaz para las aplicaciones del tipo del proveedor.

La transferencia garantiza que pueda realizarse el proceso de actualización, pues a través de ella los paquetes de actualización se desplazan desde el servidor de actualizaciones hacia la aplicación actualizadora, que actuaría a su vez sobre el servidor de aplicaciones del Sistema alas HIS. El proceso del restablecimiento de la versión anterior asegura en caso de fallo que el sistema retorne a su estado anterior para no afectar así a los usuarios de la aplicación.

Para confeccionar un actualizador que cumpla con las necesidades del alas HIS se deberían tener en cuenta algunas acciones como: la gestión de versiones; la realización peticiones al servidor de actualizaciones en busca de una nueva versión existente; la transferencia de los paquetes contenedores de la actualización; la creación de una copia del sistema existente para ser reutilizado si la actualización no fuese efectiva y finalmente mantener un registro de todas la operaciones que se realizaran para garantizar un control eficiente del proceso de actualización.

Por lo que para la implementación de la Infraestructura de Actualización para el Sistema de Información Hospitalaria alas HIS se propone utilizar las ventajas que ofrecen cada uno de estos tipo de actualizadores, siendo además el tipo de actualizado: Productos Actualizadores del Proveedor la principal guía para la solución del problema.

1.2. Sistemas automatizados existentes

Se realizó un estudio de los sistemas automatizados existentes que tengan entre sus objetivos fundamentales la actualización de aplicaciones informáticas. Con el objetivo de encontrar soluciones que

puedan ser utilizadas para resolver el problema planteado. Se describieron las principales características y desventajas de estos sistemas, lo cual ofreció así mayor información para la construcción posterior de una solución óptima.

1.2.1. Repositorios de actualizaciones:

Symantec LiveUpdateServer:

Es un servidor de actualizaciones de la empresa Symantec donde se almacenan actualizaciones referentes a los productos de esta compañía. Recibe una lista de peticiones desde el Symantec LiveUpdateClient y si estas están disponibles son descargadas hacia los clientes donde se realiza la actualización. (6)

Ventajas:

- ✓ Mantiene un control sobre el proceso de descarga de actualizaciones desde los clientes, con el fin de informar si existió algún error.

Desventajas:

- ✓ Solo almacena actualizaciones para productos de la compañía Symantec.
- ✓ Expone las actualizaciones sobre el protocolo HTTP que no garantiza la seguridad de la descarga.
- ✓ No dispone de ningún mecanismo que verifique la integridad de los archivos descargados.

Proveedor de Actualización del Sistema de Gestión Penitenciaria Venezolano (SIGEP):

Es el encargado de notificar y servir las actualizaciones para el Actualizador del Sistema de Gestión Penitenciaria Venezolano. Cuando se encuentra disponible una nueva actualización su función principal es la de informar a todos sus clientes que ya está en existencia. (7)

Ventajas:

- ✓ Garantiza la integridad del paquete de actualización a través de un algoritmo resumen.

Desventajas:

- ✓ Necesita llevar un control de las direcciones de sus clientes por lo que si alguna cambia este cliente queda fuera del alcance del servidor.

1.2.2. Aplicaciones actualizadoras:

Symantec LiveUpdateClient:

Es un actualizador de la empresa Symantec, diseñado para mantener actualizados los diferentes productos que son comercializados por esta compañía lo que lo convierte en un actualizador del tipo Productos Actualizadores del Proveedor. Esta aplicación crea una lista de los productos Symantec que estén instalados, además de otros datos referentes al cliente que necesita actualizaciones; una vez completada hace una petición al servidor Symantec LiveUpdateServer desde donde descarga los paquetes para completar la actualización. (6)

Ventajas:

- ✓ Soporta una conexión directa desde el servidor de la red local, sin necesidad de conectar individualmente cada ordenador, por lo que se ahorra tiempo y ancho de banda.
- ✓ La descarga de este programa es totalmente gratis para cualquier usuario interesado.

Desventajas:

- ✓ Solo brinda soporte de actualización a las aplicaciones creadas por Symantec.
- ✓ No está disponible para múltiples plataformas.
- ✓ Aunque es distribuida gratuitamente es un producto privativo.

Actualizador SIGHO:

Es una aplicación que está diseñada para realizar la actualización del Sistema de Información para la Gerencia Hospitalaria SIGHO por lo que es un actualizador del tipo Productos Actualizadores del Proveedor. Funciona como una herramienta que facilita el proceso de actualización del SIGHO, el cual

guía al usuario mediante una interfaz que le permitirá realizar la sustitución de los antiguos archivos del sistema. (8)

Ventajas:

- ✓ Posee un mecanismo de autenticación que evita que personal ajeno realice modificaciones que puedan poner en riesgo el funcionamiento del sistema.

Desventajas:

- ✓ Solo facilita la actualización para el SIGHO.
- ✓ El paquete de actualización debe ser copiado manualmente en el cliente donde se encuentra instalado el SIGHO.
- ✓ No es posible actualizar varios clientes a la vez.

Actualizador del Sistema de Gestión Penitenciaria Venezolano:

Esta aplicación es la encargada de escuchar los eventos que provienen del Proveedor de Actualización del Sistema de Gestión Penitenciaria Venezolano y realizar actividades en el cliente que garanticen que la actualización enviada sea instalada y activada correctamente por lo que se encuentra dentro de los Productos Actualizadores del Proveedor. (6)

Ventajas:

- ✓ Cuenta con un visualizador de Información que muestra información sobre el estado de la actualización.
- ✓ Realiza operaciones de restablecimiento de la versión anterior en caso de que falle el proceso de actualización.
- ✓ Es multiplataforma.

Desventajas:

- ✓ Solo facilita la actualización del Sistema de Gestión Penitenciaria Venezolano.

- ✓ Necesita detener e iniciar el servidor de aplicaciones para que los nuevos cambios sean puestos en práctica.

1.2.3. Análisis de los actualizadores:

Las aplicaciones que fueron analizadas anteriormente no constituyen una solución factible para implementar el proceso de actualización del Sistema de Información Hospitalaria alas HIS, principalmente debido a que son aplicaciones del tipo Productos Actualizadores del Proveedor, lo que trae consigo que solo facilite el proceso de actualización a los sistemas para los cuales han sido creados. Además en algunos casos, carecen de funcionalidades que deben tenerse en cuenta a la hora de construir una aplicación cuyo objetivo fundamental sea la actualización de un sistema.

Es por ello que las implementaciones antes vistas no tienen relevancia para su uso como solución, aunque fueron importantes como base de conocimientos para la creación de la Infraestructura de Actualización para el Sistema de Información Hospitalaria alas HIS.

1.3. Arquitectura de Software

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, es un conjunto de decisiones significativas sobre la organización de un sistema de software; la composición de esos elementos estructurales y de comportamiento en subsistemas progresivamente mayores; el estilo arquitectónico que guía esta organización: los elementos estáticos y dinámicos y sus interfaces, sus colaboraciones y su composición.

1.3.1. Middleware Orientado a Mensajes

Es una tecnología que conecta sistemas separados en una red mediante la realización y distribución de mensajes. Los mensajes pueden contener datos, instrucciones de software o ambos. La infraestructura MOM o Middleware Orientado a Mensajes se basa típicamente en un sistema de colas que almacena los mensajes pendientes de entrega y hace un seguimiento de cada mensaje hasta que ha sido entregado. Permite distribuir aplicaciones a través de plataformas heterogéneas, y reduce la complejidad del

desarrollo de aplicaciones que abarcan múltiples sistemas operativos y protocolos de red. API que se extienden por diversas plataformas y redes suelen ser proporcionados por MOM. (9)

MOM consta de dos posibilidades arquitectónicas:

- ✓ Arquitectura centralizada:

Se basa en un servidor de mensajes (también conocido como router de mensajes), que permite la comunicación de aplicaciones con bajo acoplamiento, haciéndolas independientes entre sí.

- ✓ Arquitectura descentralizada:

Implementa la transmisión multicast/ broadcast lo cual aumenta la complejidad del cliente, ya que recibe información local periódica (como suscripciones y autorizaciones) acerca de otros clientes.

La tecnología MOM puede ser de 2 modelos:

- ✓ Mensajería punto a punto:

Es una mensajería de Petición/ Respuesta, donde el Emisor realiza una petición, la misma se inserta en la Cola de Peticiones, el Receptor la saca de esta cola y genera una respuesta que es enviada a la Cola de Respuestas, de donde el Emisor deberá sacarla.

- ✓ Mensajería por difusión (publicación/suscripción):

Es una mensajería mediante publicación/suscripción donde intervienen productores y consumidores. Los consumidores se suscriben a un asunto o tema, los productores publican mensajes en un tema determinado y los mensajes sobre un tema se envían a todos los consumidores suscritos.

1.3.2. Cliente-Servidor

La modalidad o arquitectura Cliente-Servidor es aquella en la que confluyen una serie de aplicaciones basadas en dos categorías que cumplen funciones diferentes (una requiere servicios y la otra los brinda). Pueden realizar tanto actividades en forma conjunta como independientemente. Esas dos categorías son justamente cliente y servidor. El cliente es aquel que requiere un servicio del servidor. En esta categoría se realizan funciones de software basándose en el hardware, pero en caso de no tener la capacidad de procesar los datos necesarios, recurre al servidor y espera a que este le brinde los servicios solicitados. El cliente es una estación de trabajo o computadora que está conectada a una red a través de la cual puede acceder al servidor. (10)

Por el contrario, el servidor es la máquina desde la que se suministran servicios y que está a la espera del requerimiento del cliente. Una vez hecho, busca la información solicitada y le envía la respuesta al cliente; incluso puede enviar varios servicios a la vez, lo que es posible porque entre ellos están conectados mediante redes LAN[2] o WAN[3].

Entre las características fundamentales de esta arquitectura se encuentra que el cliente y el servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, por lo cual realizan actividades o tareas independientes, las funciones de cliente y servidor pueden estar en plataformas separadas, o en la misma plataforma, un servidor da servicio a múltiples clientes en forma concurrente, cada plataforma puede ser escalable independientemente y que los cambios realizados en las plataformas de los clientes o de los servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

1.4. Metodología de desarrollo

1.4.1. Proceso Unificado de Desarrollo

[2] Por sus siglas en inglés red de área local.

[3] Por sus siglas en inglés red de área global.

El Proceso Unificado de Desarrollo (Racional Unified Process RUP por sus siglas en inglés) es una metodología adaptable al contexto y necesidades de cada organización cuyo fin es entregar un producto de software. Durante su puesta en práctica se estructuran todos los procesos y se mide la eficiencia de la organización. Además es un proceso de desarrollo de software el cual utiliza el lenguaje unificado de modelado UML. Constituye así la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos**.** RUP se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. (11)

Principales características de RUP:

- ✓ Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- ✓ Pretende implementar las mejores prácticas en Ingeniería de Software.
- ✓ Desarrollo iterativo.
- ✓ Administración de requisitos.
- ✓ Uso de arquitectura basada en componentes.
- ✓ Control de cambios.
- ✓ Modelado visual del software.
- ✓ Verificación de la calidad del software.

RUP divide el proceso de desarrollo en ciclos, por lo que se obtiene un producto final al culminar cada uno de ellos, que a la vez se dividen en fases:

- ✓ Concepción.
- ✓ Construcción.
- ✓ Transición.
- ✓ Mantenimiento.

En RUP se definen nueve flujos de trabajo distintos, separados en dos grupos. Los flujos de trabajo de ingeniería son los siguientes:

- ✓ Modelado del negocio.
- ✓ Requisitos.
- ✓ Análisis y diseño.
- ✓ Implementación.
- ✓ Pruebas
- ✓ Despliegue.

Los flujos de trabajo de apoyo son:

- ✓ Administración del proyecto.
- ✓ Configuración y control de cambios.
- ✓ Entorno.

1.4.2. Lenguaje Unificado de Modelado

El lenguaje para modelado unificado (Unified Modeling Language por sus siglas en inglés UML), es utilizado para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Se utiliza para definir un sistema, detallar los artefactos en el sistema, documentar y construir, es el lenguaje en el que está descrito el modelo. Se puede aplicar en el desarrollo de software, lo que entrega gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar. (12)

UML prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos y describe la semántica esencial de lo que estos diagramas y símbolos significan. Entre más complejo es el sistema que se desea crear más beneficios presenta el uso de UML, por las siguientes razones:

Los principales beneficios de UML son:

- ✓ Mejores tiempos totales de desarrollo (de 50 % o más).
- ✓ Modelar sistemas utilizando conceptos orientados a objetos.
- ✓ Establecer conceptos y artefactos ejecutables.
- ✓ Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- ✓ Crear un lenguaje de modelado utilizado por humanos y por máquinas.
- ✓ Mejor soporte a la planificación y al control de proyectos.
- ✓ Alta reutilización y minimización de costos.

1.5. Tecnologías de Desarrollo

1.5.1. Java

Java es toda una tecnología orientada al desarrollo de software con la cual se puede realizar cualquier tipo de programa. Hoy en día, la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su plataforma J2EE (Java 2 Enterprise Edition). La tecnología Java está compuesta básicamente por 2 elementos: el lenguaje Java y su plataforma. Algunas de sus características más notables son: robustez, gestiona la memoria automáticamente, multiprocesos, cliente-servidor y mecanismos de seguridad incorporados. (13)

Java tiene ventajas significativas sobre otros lenguajes y entornos que lo hacen apto para cualquier tarea de programación entre ellas se encuentran:

- ➢ Fácil de aprender:

Java fue diseñado para ser fácil de usar y es por tanto fácil de escribir, compilar, depurar y aprender que otros lenguajes de programación.

- ➢ Es orientado a objetos:

Esto le permite crear programas modulares y código reutilizable.

- Es independiente de la plataforma:

Una de las ventajas más importantes de Java es su capacidad para moverse fácilmente de un sistema informático a otro. La capacidad para ejecutar el mismo programa en diferentes sistemas es crucial para el éxito de un software.

- Es distribuido:

Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.

- Es seguro:

Java considera la seguridad como parte de su diseño. El lenguaje Java, el compilador, intérprete, y el entorno de ejecución de cada uno se desarrolló con la seguridad en mente.

- Java es robusto:

Exhaustivo de la fiabilidad. Java pone mucho énfasis en el control temprano de posibles errores, compiladores de Java son capaces de detectar muchos problemas que aparecen durante el tiempo de ejecución en otros lenguajes.

- Java es multiproceso:

Multiproceso es la capacidad de un programa para realizar varias tareas al mismo tiempo dentro de un programa. En Java, la programación multiproceso se ha integrado progresivamente, mientras que en otros lenguajes, procedimientos específicos tienen que ser llamados para implementar múltiples subprocesos.

Java Enterprise Edition 5 (Java EE 5)

Es una plataforma de programación, parte de la Plataforma Java, para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java con arquitectura de n capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones

Java EE. Considerada informalmente como un estándar debido a que los proveedores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por The Java Community Process (JCP por sus siglas en inglés). (14)

1.5.2. Seam Framework

JBoss Seam es una nueva y potente infraestructura para desarrollar aplicaciones Web 2.0, al unificar e integrar tecnologías como AJAX, JSF[4], EJB 3[5], Java Portlets, Business Process Management, Drools, Hibernate y Java Persistence API en una única solución. El desarrollo de Seam, utiliza JSF y EJB es muy ágil ya que reduce el nivel de configuración necesario para la integración y aprovecha al máximo las ventajas de cada una de las tecnologías, lo que hace al proyecto más estable y legible. (15)

Entre las principales ventajas de este marco de trabajo se puede encontrar:

- Fuertemente basado en anotaciones (elimina XML extensos) con una gran cantidad de las mismas que permiten las configuraciones más variadas.
- Gran cantidad de tipos de componentes: Interceptores, Beans, Logger, manejadores de eventos, FacesMessages, IdentityStore.
- Bijection: Permite la clásica inyección y agrega la outyección de objetos.
- Componentes visuales: Permite utilizar frameworks de componentes visuales como RichFaces o ICEFaces.
- JPA: integración directa con JPA. Inyección del EntityManager.
- Seguridad: Presenta un modelo de autenticación y autorización completo basado en clases y anotaciones. Extensible pues usa componentes Seam.

[4] Java Server Faces

[5] Enterprise Java Beans

- SpringFramework: Integración con Spring lo que permite inyectar en componentes Seam Spring-beans y definir Spring-beans como componentes Seam.

1.5.3. Hibernate Framework

Hibernate es una herramienta para la plataforma Java que facilita el mapeo de entidades entre una base de datos relacional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones. Es una herramienta ORM (Object Relational Mapping) o Mapeo Objeto-Relacional que destaca entre sus ventajas una buena documentación y estabilidad. Tiene gran alcance, alto rendimiento, persistencia relacional y servicio de consulta. Permite desarrollar clases persistentes que sigan el lenguaje orientado a objetos, incluyendo asociación, herencia, polimorfismo, la composición y las colecciones. (16)

Entre las principales ventajas que se pueden encontrar en Hibernate están:

- Productividad: evita código confuso en la capa de persistencia, lo que permite centrarse en la lógica de negocio.
- Mantenibilidad: por tener pocas líneas de código permite que el código sea más claro. Al dividir la capa de persistencia se pueden identificar los errores muy fácilmente.
- Rendimiento: hibérnate tiene un buen desempeño pero todo depende realmente de como se realicen las consultas y como se configure el Framework.
- Independencia del proveedor: una solución ORM te abstrae del SGBD. Permite desarrollar localmente con bases de datos ligeras sin implicación en el entorno de producción.

1.5.4. JPA (Java Persistence API)

Java Persistence API, más conocida por sus siglas JPA, es la API de persistencia desarrollada para la plataforma Java EE, incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional, es decir, la relación entre entidades Java y tablas de la

base de datos, lo que se realiza mediante anotaciones en las propias clases de entidad, por lo que no se requieren ficheros descriptores XML. Además pueden definirse transacciones como anotaciones JPA. (17)

1.5.5. Librerías True License

True License incluye un conjunto de librerías que permiten entre otras cosas crear, instalar, verificar licencias; trabaja con Java Cryptography Extension (JCE) y Java Security API lo que permite cifrar la extensión de la aplicación, trabajar con licencias a partir de Key Stores y Certificados.

Beneficios que reporta:

- ✓ Protege la inversión y propiedad intelectual en el desarrollo del software.
- ✓ Quién quisiera violar la licencia necesitaría acceder a niveles múltiples de codificaciones, asegurar firmas digitales, códigos y cheques de validación.

1.5.6. JAX- WS

Es un API de Java para la creación de servicios web. Es parte de la plataforma Java EE de Sun Microsystems. Al igual que los otros API de Java EE, JAX-WS utiliza anotaciones, es introducido en Java SE 5 para simplificar el desarrollo y el despliegue de los clientes de servicios web. La implementación de referencia de JAX-WS es desarrollada en código abierto y es parte del proyecto GlassFish. (18)

1.6. Tecnologías de intercambio.

1.6.1. Servicio Web

Existen múltiples definiciones sobre lo que son los servicios web lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican; se habla de ellos como un conjunto de aplicaciones o de tecnologías con capacidad para interactuar en la Web, las cuales intercambian datos entre sí con el objetivo de ofrecer servicios que proporcionen mecanismos de comunicación estándares. (19)

Los Servicios Web cuentan con ventajas como:

- ✓ Se basan en protocolo de transferencia de hipertexto HTTP[6] sobre el protocolo de control de transmisiones TCP[7] en el puerto 80.
- ✓ Proporciona interoperabilidad entre aplicaciones de software que se ejecutan en plataformas distintas.
- ✓ Aporta gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo, en uno, no deben afectar al otro.
- ✓ Permiten la lógica de negocio de muchos sistemas diferentes para ser expuestos en la web. Esto da a sus aplicaciones la libertad de elegir los servicios web que ellos necesitan, desarrollarlos nuevamente para cada cliente.
- ✓ Permiten fácilmente combinar software y servicios de diferentes empresas y lugares para conformar un nuevo y mejorado servicio integral.

1.6.2. SFTP

SFTP[8] es un protocolo de red que proporciona la funcionalidad necesaria para la transferencia y manipulación de archivos sobre un flujo de datos fiable. Se utiliza comúnmente el Protocolo de Transferencia de Archivos FTP[9] sobre SSH[10], para proporcionar la seguridad a los datos, aunque permite ser usado con otros protocolos de seguridad. Por lo tanto, la seguridad no la provee directamente el protocolo FTP, sino el SSH o el protocolo que sea utilizado en su caso para este cometido. (20)

1.7. Herramientas

1.7.1. Visual Paradigm para UML 6.4

[6] Hypertext Transfer Protocol.

[7] Transmission Control Protocol.

[8] Secure File Transfer Protocol.

[9] File Transfer Protocol.

[10] Secure Shell.

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida del desarrollo de software completo: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (21)

1.7.2. Eclipse 3.4 Ganymede

Se define como un IDE de desarrollo, Eclipse es únicamente un armazón sobre el que se pueden montar herramientas de desarrollo para varios lenguajes, mediante la implementación de los plugins adecuados. La arquitectura de plugins de Eclipse permite introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces y ayuda en línea para librerías. Todas las versiones de Eclipse necesitan tener instalado en el sistema una máquina virtual Java (JVM), preferiblemente JRE[11] o JDK[12] de Sun Microsystems. Eclipse se distribuye bajo licencia EPL[13]. (22)

1.7.3. PgAdmin III como administrador de base de datos

PgAdmin es una aplicación gráfica para trabajar con el gestor de bases de datos PostgresSQL, usa la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usan en Linux, FreeBSD, Solaris, Mac OS X y Windows.

La aplicación incluye también un editor SQL con resaltado de sintaxis, un editor de código en el servidor y un agente para lanzar script programados. Está diseñado tanto para escribir consultas SQL como para

[11] Java Runtime Environment.

[12] Java Developer Kit.

[13] Eclipse Public License.

desarrollar bases de datos complejas. La conexión al servidor puede hacerse mediante TCP/IP o Unix Domain Sockets (en plataformas Unix) y es encriptado mediante SSL[14]para mayor confidencialidad. (23)

1.7.4. JBoss Application Server 4.2

JBoss es el servidor de aplicaciones de código abierto que lidera la comunidad actual. Ofrece grandes ventajas en lo que se refiere a escalabilidad del sistema y utilidades de funcionamiento, al mismo tiempo que proporciona un entorno robusto y rápido para la ejecución de las aplicaciones. (24)

En la siguiente tabla se muestran sus principales ventajas:

Características	JBoss 4.2	Geronimo 2	Tomcat 6	GlassFish 2
Soporte para Java EE 5	Parcial	Si	No	Si
Soporte para EJB 3.0	Si	Si	Disponible	Si
Soporte para JSP 2.1 y 2.5	Si	Si	Si	Si
Soporte para Java Server Faces 1.2	Si	Si	Disponible	Si
Soporte para plug-in personalizados	Si	Si	No	-
Soporte para motor de reglas del negocio	Disponible	Disponible	Disponible	Disponible
Soporte para Hibernate 3.x	Si	Disponible	Disponible	Si
Soporte para JBoss Seam	Si	Si	Disponible	Si
Soporte para cauterización	Si	Si	Parcial	Si
Soporte para conexión con Eclipse IDE	Si	Si	Si	Si

1.1. Tabla de comparación entre JBoss Application Server 4.2 y otras aplicaciones similares características

[14]Secure Socket Layer.

JBoss es el encargado de ejecutar aplicaciones Java en el servidor y devolver los datos correspondientes a las peticiones realizadas por el usuario. Permite crear, implementar, integrar, organizar y presentar aplicaciones y servicios web en una arquitectura orientada a servicios.

1.7.5. NetBeans 6.9

Es una plataforma de desarrollo creada fundamentalmente para el trabajo con el lenguaje de programación Java. Las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. Es un proyecto de código abierto donde se integra un ambiente de desarrollo para desarrolladores del software. Soporta lenguajes como Java, C/C + +, PHP, Java Script y el Rubí. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. (25)

1.7.6. PostgreSQL 9.0

PostgreSQL es un poderoso sistema de base de datos relacionales. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que le ha hecho ganar una gran reputación por la integridad fiabilidad y exactitud de los datos. Se puede ejecutar en diferentes sistemas operativos tales como Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Macintosh OS X, Solaris, Tru64), y Windows. Está publicado bajo la licencia BSD[15]. Utiliza el lenguaje SQL para llevar a cabo búsquedas de información. Soporta llaves foráneas, joins, vistas, triggers y métodos almacenados. Incluye varios tipos de datos además se pueden almacenar grandes objetos binarios incluyendo imágenes, sonidos y videos. (26)

1.7.7. OpenSSH Server 5.0

OpenSSH es una versión gratis de las herramientas de conectividad SSH. Cifra todo el tráfico (incluyendo contraseñas) para eliminar efectivamente espionaje, robo de conexiones y otros ataques para ello usa el protocolo SSH. Además ofrece la creación de canales seguros y varios métodos de autenticación, además soporta todas las versiones del protocolo SSH. Es desarrollado como parte de la seguridad proyecto OpenBSD. (27)

[15] Berkeley Software Distribution.

En el presente capítulo se realizó un estudio de la fundamentación teórica y del conjunto de herramientas, metodologías y tecnologías de intercambio actuales que formarán parte de la propuesta de Infraestructura de Actualización para el Sistema de Información Hospitalaria alas HIS. Además se investigó sobre algunos sistemas cuya función principal es realizar el proceso de actualización ya sea que funcionen como aplicación actualizadora o como servidores de actualización; los cuales no solucionan las necesidades planteadas pero si forman parte de un elemento valorativo que aportó ideas para la construcción de la futura solución.

Capítulo 2: DESCRIPCIÓN DE LA ARQUITECTURA

En el presente capítulo se exponen los requisitos no funcionales encontrados durante el flujo de trabajo de requerimientos y el modelo del dominio principalmente, cuyo objetivo es documentar y explicar el problema definido. Queda descrita y fundamentada la arquitectura a utilizar en la solución del problema antes planteado, además se modela la vista de despliegue y se realiza un estudio acerca de la seguridad. Finalmente se abordan los estándares de codificación a utilizar durante la implementación.

2.1. Modelo del dominio

El modelo de dominio es utilizado para brindar una representación visual del problema o del proyecto, a través de objetos del dominio o clases, cuando se carecen de los elementos necesarios para modelar un negocio.

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” o los eventos que suceden en el entorno en el que trabaja el sistema.

2.1.1. Conceptos fundamentales del modelo de dominio

Los conceptos fundamentales del modelo de dominio no son más que los objetos o clases que mostrarán como se comportará el sistema, para logara un mejor entendimiento del mismo.

Se han identificado las siguientes clases:

- ✓ Módulos: contiene todos los módulos por los que están compuestos el sistema de Información Hospitalaria alas HIS.
- ✓ Línea Base: contiene toda la información referente a la línea base.
- ✓ Base Datos: contiene toda la información referente al HIS, los paquetes de actualizaciones, los módulos y las relaciones entre ellos.
- ✓ Versión _HIS: contiene todas las versiones de los HIS existentes en el servidor de actualizaciones que están disponibles para realizar la actualización.
- ✓ Base de Datos_Versiones: contiene todas las versiones de los HIS existentes.
- ✓ Servidor de Actualizaciones: contiene todas las actualizaciones de los HIS.
- ✓ Petición de Actualización: es proceso donde los clientes de actualización solicitan actualización para su sistema.
- ✓ Aplicación Actualizadora: representa la aplicación encargada de realizar el proceso de actualización en el HIS.
- ✓ Actualización _HIS: es el compactado de actualización que es descargado desde el servidor de actualizaciones hacia la aplicación actualizadora.
- ✓ Mensaje Efectividad _HIS: representa un mensaje que confirma que el proceso de actualización fue satisfactorio.
- ✓ Mensaje Fallo _HIS: representa un mensaje que confirma que el proceso de actualización no se efectúo con éxito.
- ✓ Sistema a Actualizar: representa el servidor de aplicaciones del Sistema de Información Hospitalaria alas HIS y su servidor de base de datos.

- ✓ Copia _HIS: contiene todos datos pertenecientes al sistema que se encontraba funcionamiento antes de comenzar el proceso de actualización.
- ✓ Descripción_HIS: contiene toda la información referente al sistema que está desplegado, las versiones de los módulos, versión de la base de datos, versión de la línea base y la versión del sistema. Además de contener el archivo de tipo licencia del sistema.
- ✓ Fichero de Configuración: contiene los datos referentes a la fecha y la hora en que se realizará la actualización, puerto y contraseña para conectarse al HIS, nombre del sistema, dirección del servidor de aplicaciones.
- ✓ Licencia_HIS: Contiene los datos referentes a la licencia, el tiempo de duración de la misma, el sistema al que pertenece, el serial que compone la licencia.

2.1.2. Diagrama del modelo de dominio

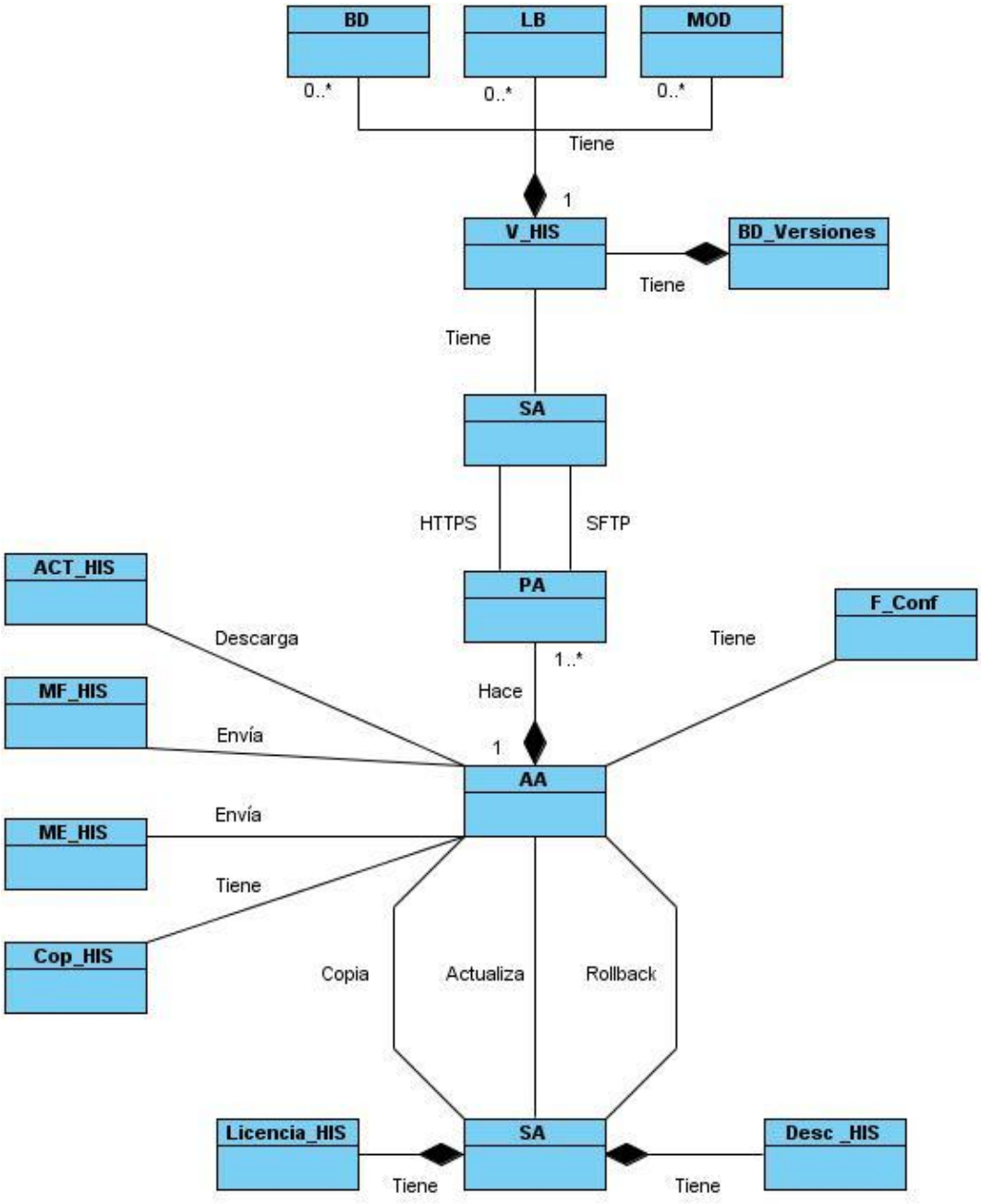

Figura 3 - Modelo de dominio de Infraestructura de Actualización

2.2. Requisitos no funcionales.

Los requisitos de manera general son las condiciones o capacidades que tienen que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

Los requisitos no funcionales dentro del sistema son aquellos que describen las propiedades o cualidades que el producto debe tener. Son las características que hacen al producto atractivo, usable, rápido o confiable, en muchos casos los requisitos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requisitos funcionales, es decir una vez se conozca lo que el sistema debe hacer se determina cómo ha de comportarse y qué cualidades debe tener.

2.2.1. Requisitos de Software

Los requisitos de software hacen mención a todo el software entiéndase como programas de los cuales el sistema debe disponer.

RNF1 Estaciones de administración: requiere un navegador web, que puede ser IE 7, Opera 9, Google chrome 1 y Firefox 2 o versiones superiores de estos.

RNF2 Servidor de base de datos: sistema operativo Linux o Windows y el gestor de base de datos PostgreSQL 8.3 o superior.

RNF3 Servidor de actualización: sistema operativo Linux o Windows, el servidor de aplicaciones JBoss 4.2x y la Máquina Virtual de Java (JVM) 1.6 o superior.

RNF4 Servidor de ficheros SFTP: sistema operativo Linux o Windows y el servidor Open SSH Server 1.5 o superior.

RNF5 Servidor de aplicación actualizadora: sistema operativo Linux y la Máquina virtual de Java (JVM) 1.6.

2.2.2. Requisitos de Hardware:

Los elementos de hardware corresponden a todas las partes físicas y tangibles de una computadora, como parte de los requisitos de hardware están aquellas partes físicas imprescindibles para el uso del sistema.

RNF6 Estaciones de administración: procesador Intel Dual-Core con velocidad de 2 GHz y 256 MB de memoria RAM.

RNF7 El servidor de base de datos: procesador Intel Dual-Core con velocidad de 3 GHz, 1 GB de memoria RAM y disco duro de 250 GB.

RNF8 El servidor de actualizaciones tanto como el de ficheros SFTP: procesador Intel Dual-Core con velocidad de 3 GHz, 4GB de Memoria RAM, disco duro de 250 GB.

RNF9 Servidor de aplicación actualizadora: Ordenador Pentium IV, 1GB de Memoria RAM y disco duro de 10 GB.

2.3. Seguridad

La seguridad del sistema quedará implementada a fin de conseguir que no sea violada la disponibilidad, la confidencialidad y la integridad de la información que será manejada. Antes de responder cualquier petición de actualización se comprobará la autenticidad del HIS por medio de una llave única que tendrá asignada. Debido a que la información que se manejará entre el servidor de actualizaciones y el servidor de aplicación actualizadora no debe seguir un destino desconocido, se garantizará que la misma se traslade por un canal seguro a través del protocolo SFTP. La conexión desde el servidor de aplicación actualizadora hacia el servidor de archivos SFTP estará protegida por contraseña y como la información que se maneja es sensible y solo debe ser accedida por el personal autorizado se precisa un control de acceso a nivel de usuarios y contraseñas.

2.4. Descripción de la arquitectura

2.4.1. Middleware Orientado a Mensajes (MOM)

La arquitectura definida guiará el desarrollo del sistema y de esta dependerá en gran medida el éxito de la solución, por lo que la misma debe ser escogida consiente y consecuentemente. Por las características de la infraestructura a desarrollar se hace necesario combinar varias arquitecturas que abarquen los diferentes elementos que la componen seleccionándose MOM con una arquitectura centralizada, implementándose el modelo Point to Point en la comunicación entre el servidor de actualizaciones y el servidor de aplicación actualizadora lo que propociona una comunicación mediante el protocolo HTTPS[16], para aprovechar todas las ventajas que brinda el uso de servicios web.

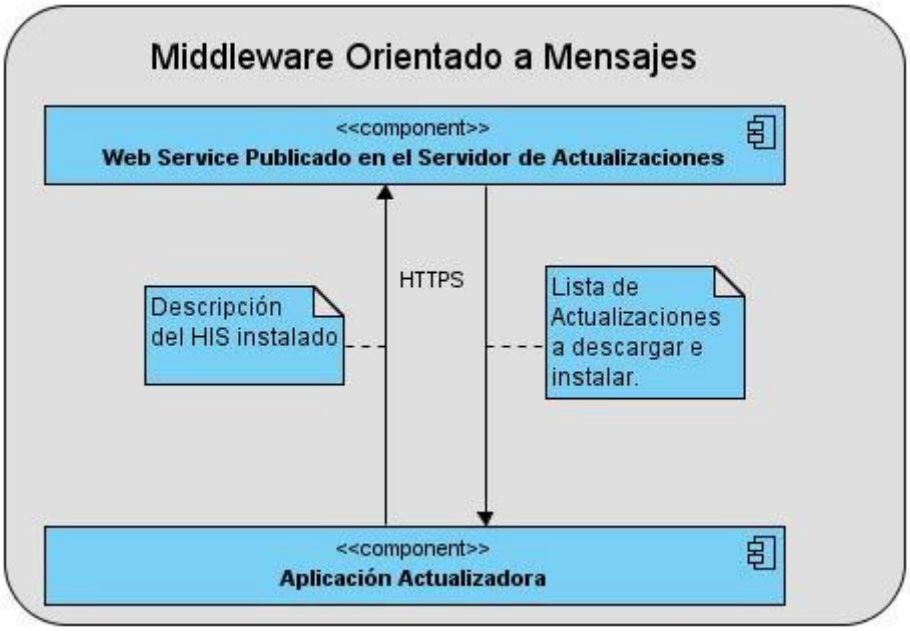

Figura 4 - MOM aplicada a la infraestructura de actualización

[16] Secure Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto segura)

2.4.2. Cliente – Servidor

La arquitectura Cliente-Servidor para la comunicación entre el servidor de ficheros y el cliente de actualización, en este caso mediante el protocolo SFTP, lo que garantiza una conexión segura, integridad de los datos además de gran rapidez en la descarga de las actualizaciones.

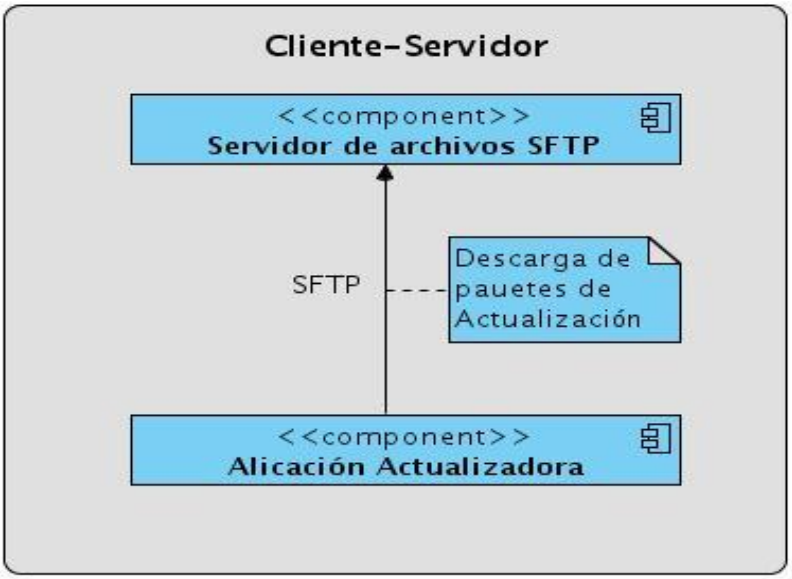

Figura 5 - Cliente servidor aplicada a la infraestructura de actualización

2.5. Vista de Despliegue.

La vista de despliegue es una representación del hardware utilizado en las implementaciones de sistemas y las conexiones existentes entre sus componentes, de esta manera es más fácil comprender en qué condiciones se piensa que el sistema pueda ejecutarse.

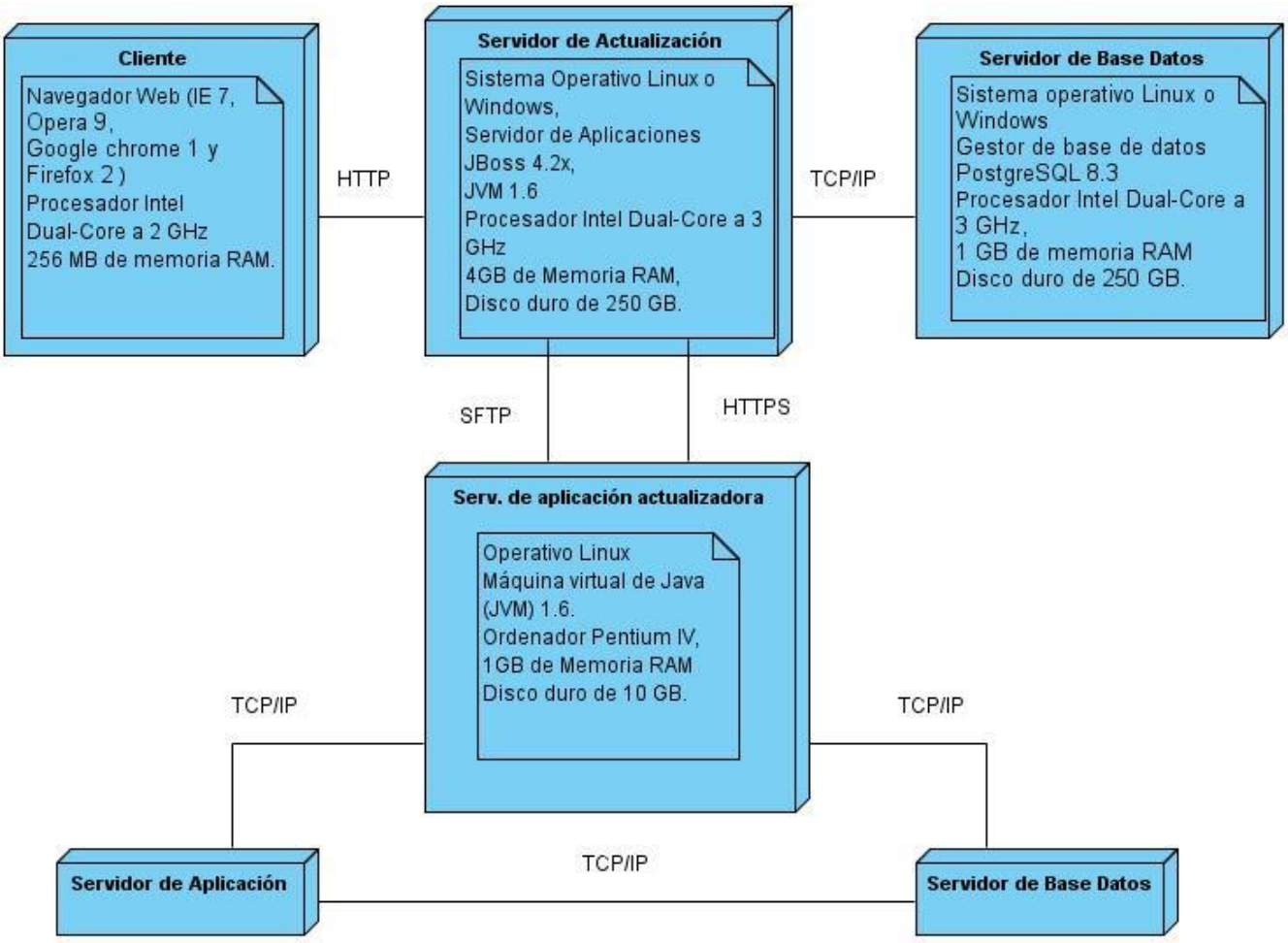

Figura 6 - Diagrama de despliegue

2.6. Estándares de codificación y estilos arquitectónicos a utilizar.

Los estándares de codificación y estilos arquitectónicos a utilizar son una serie reglas o pautas que guiarán el proceso de implementación con el objetivo de garantizar que el código generado dentro del sistema sea comprensible para el resto de los desarrolladores.

2.6.1. Bases de datos, sentencias SQL, nombre y apariencia de los campos

- ✓ Apariencia de los campos: el nombre a emplear para los campos debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo: id_serial
- ✓ Nombre de los campos: en caso de identificadores el campo identificador de cada tabla se llama: “id”, en los casos en que funcionen como llave foránea se llaman: “id _ nombre de la tabla”. Ejemplo: id_modulo
- ✓ Sentencias SQL: las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas.
- ✓ Nombre empleado para las bases de datos, las vistas, las tablas, los campos y los procedimientos almacenados: deben permitir que con sólo leerlos se conozca el propósito de los mismos.

2.6.2. Clases y objetos

- ✓ Apariencia de clases: la primera letra de los nombres de las clases debe escribirse en mayúscula y el resto del nombre en minúscula. Ejemplo: Clase_ejemplo.
- ✓ Apariencia de atributos: el nombre de los atributos de una clase debe escribirse en minúscula y la primera letra debe estar en correspondencia con el tipo de dato del atributo en específico. Ejemplo: stringsnommodulo;
- ✓ Apariencia de las funciones: los nombres de las funciones deben ser verbos que denoten la acción de la función. La primera letra del identificador se escribirá en minúscula y la primera letra de las siguientes palabras concatenadas en mayúscula. Ejemplo: buscarModulo ().

  Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set.
- ✓ Los nombres de las clases, los objetos, los atributos y las funciones: debe permitir que con sólo leerlo se conozca el propósito de los mismos. Deben ser lo más simple y sugerente posible, para lo cual se utilizan palabras completas y abreviaturas conocidas.

2.6.3. Comentarios, separadores, líneas, espacios en blanco y márgenes

- ✓ Ubicación de comentarios: es recomendable comentar al inicio de la clase o función para especificar el objetivo de la misma así como los parámetros que usa (especificar tipos de dato y objetivo del parámetro).
- ✓ Líneas en blanco: es recomendable dejar una línea en blanco antes y después de la declaración de una clase y de la implementación de una función.
- ✓ Espacios en blanco en los operadores lógicos y aritméticos: entre operadores lógicos y aritméticos es recomendable usar espacios en blanco para lograr una mayor legibilidad en el código. Ejemplo: modulo = nommodulo

2.6.4. Variables y constantes

- ✓ Apariencia de variables: se recomienda para el nombre de las variables que contenga un prefijo para el tipo de dato en minúscula. Los nombres de las variables deben ser cortos y significativos, de manera que se entienda con facilidad su significado. Se deben evitar las variables de una sola letra, excepto para las temporales de corto uso. Ejemplo: sNombreModulo.
- ✓ Apariencia de constantes: se deben declarar las constantes con todas sus letras en mayúscula.
- ✓ Nombres de las variables y constantes: el nombre empleado, debe ser fácil de comprender y permitir que con sólo leerlo se conozca el propósito de la misma.

En este capítulo quedó expuesto el modelo del dominio como punto de partida para comprensión del sistema donde se modelan las clases de mayor importancia que aportan un mejor entendimiento de la solución del problema. Se describen los requisitos no funcionales necesarios para generar un sistema de calidad ya con ellos se exponen las propiedades o cualidades que el producto debe tener. Además queda expuesta la seguridad con la cual queda garantizada la disponibilidad, integridad y fiabilidad de la información gestionada por el sistema.

Se presentó la vista de despliegue desarrollada a partir de los requerimientos no funcionales propuestos y se fundamentó la arquitectura ya propuesta en el capítulo anterior. Se concreta finalmente como quedará concebida la solución, su funcionamiento y características lo cual es de suma importancia para dar cumplimiento al objetivo general de la investigación; además se realizó un análisis de los estándares de codificación y estilos a utilizar que garantizarán una implementación clara y comprensible para cualquier programador en caso de futuros mantenimientos.

Capítulo 3: Descripción y análisis de la solución propuesta

El presente capítulo contiene el diseño propuesto por el analista, una descripción de las clases u operaciones necesarias más importantes; además de una descripción de la estructura de la base de datos conocida como modelo de datos, este capítulo se complementa con una breve valoración de las técnicas de validación propuestas.

3.1. Diseño propuesto por el analista.

3.1.1. Diagrama de clases del diseño

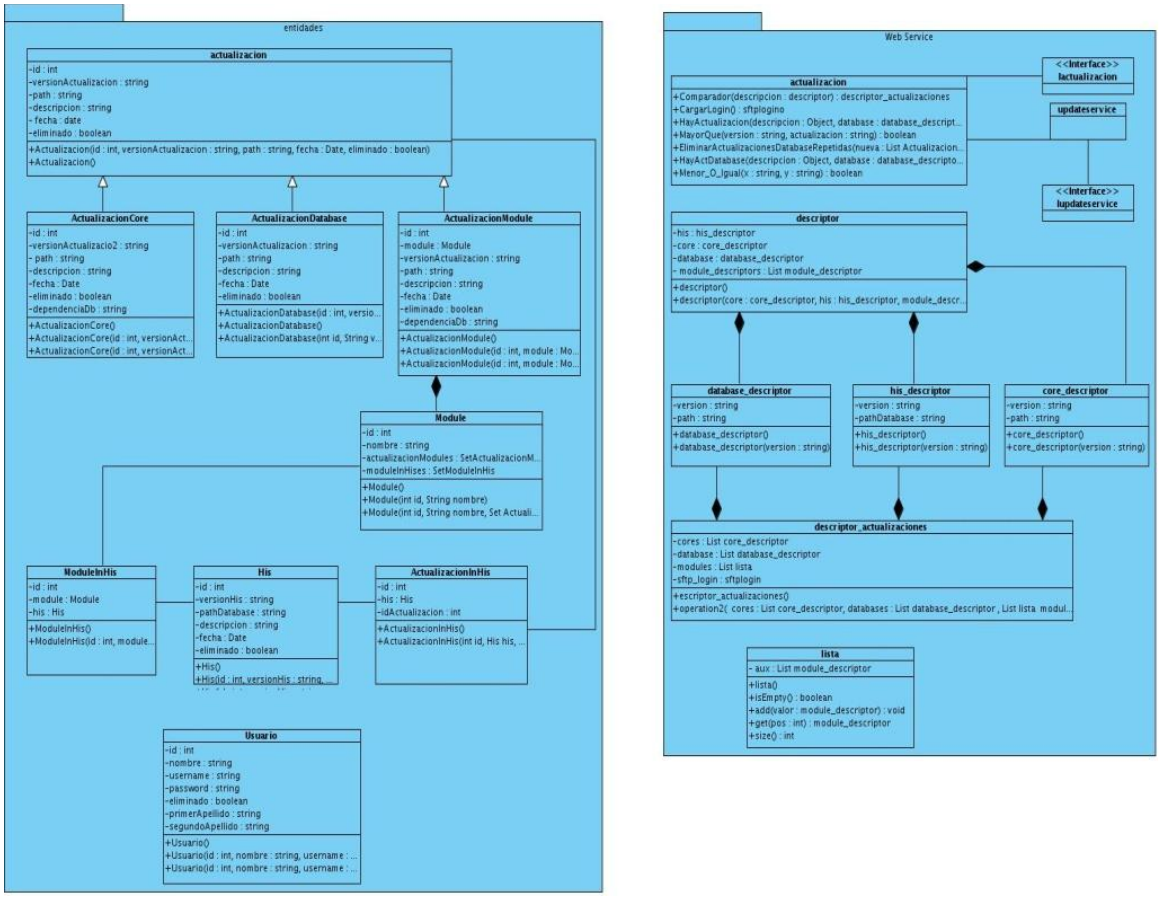

Figura 7 - Diagrama de clases: Servidor de actualizaciones

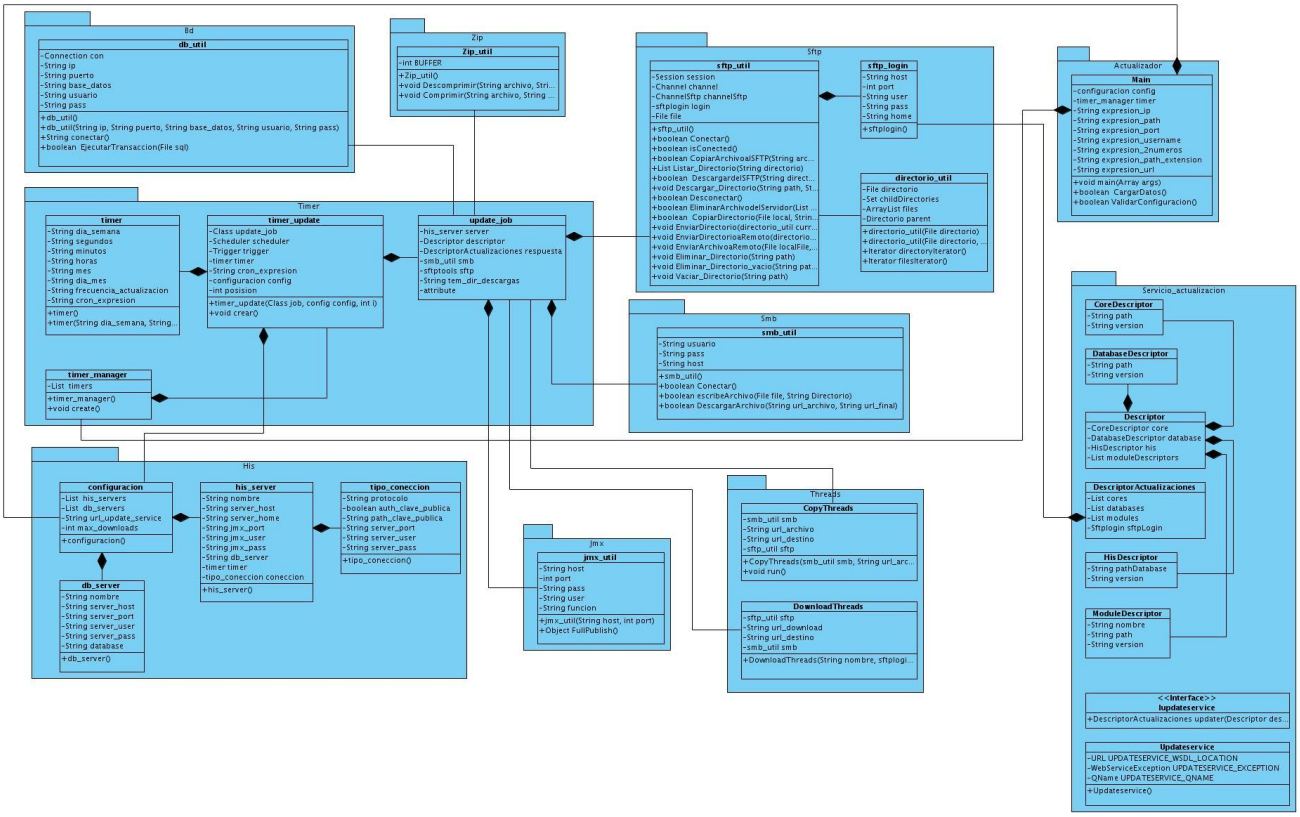

Figura 8 - Diagrama de clases: Aplicación actualizadora

3.1.2. Diagramas de interacción

Los diagramas de interacción son aquellos donde un término genérico que se aplica a varios tipos de diagramas que hacen hincapié en las interacciones entre objetos. Los diagramas de interacción tienen diferentes formas, basadas todas ellas en una misma información subyacente pero resaltando cada una un punto de vista de la misma: diagramas de secuencia, diagramas de colaboración.

3.1.2.1. Diagrama de secuencia

Un diagrama de secuencia muestra una interacción que está organizada como una secuencia temporal. En particular, muestra los objetos que participan en la interacción mediante sus líneas de vida y mediante

los mensajes que intercambian, organizados en forma de una secuencia temporal. Representa un interacción donde existe un eje de tiempo vertical que avanza a todo lo largo del diagrama y en el eje horizontal se representan los roles clasificador que muestran objetos individuales en la colaboración los cuales tiene para mostrarse una columna vertical o línea de la vida.

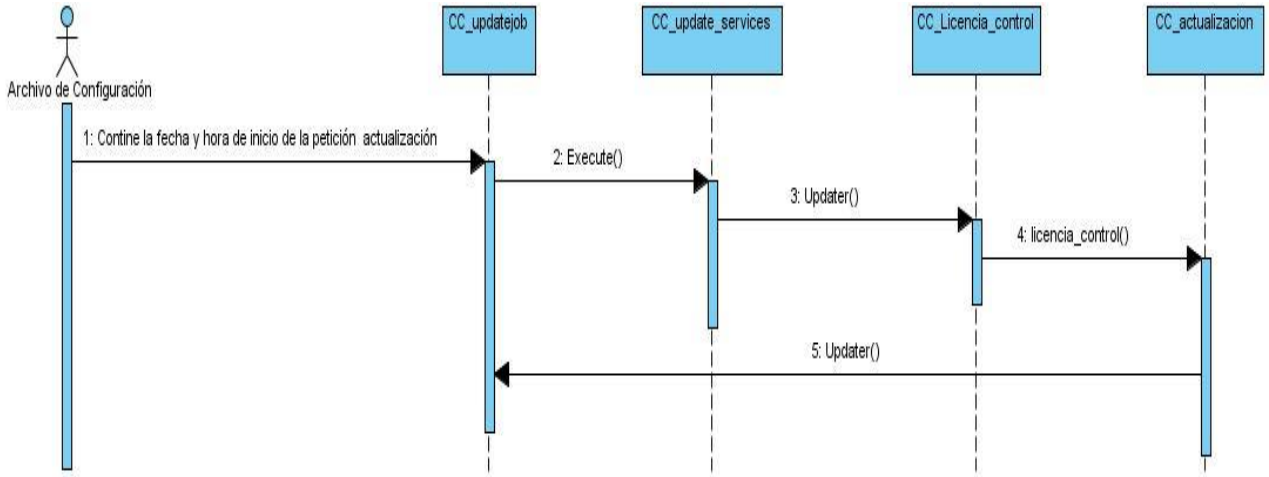

Figura 9 - Diagrama de secuencia: Petición de actualización

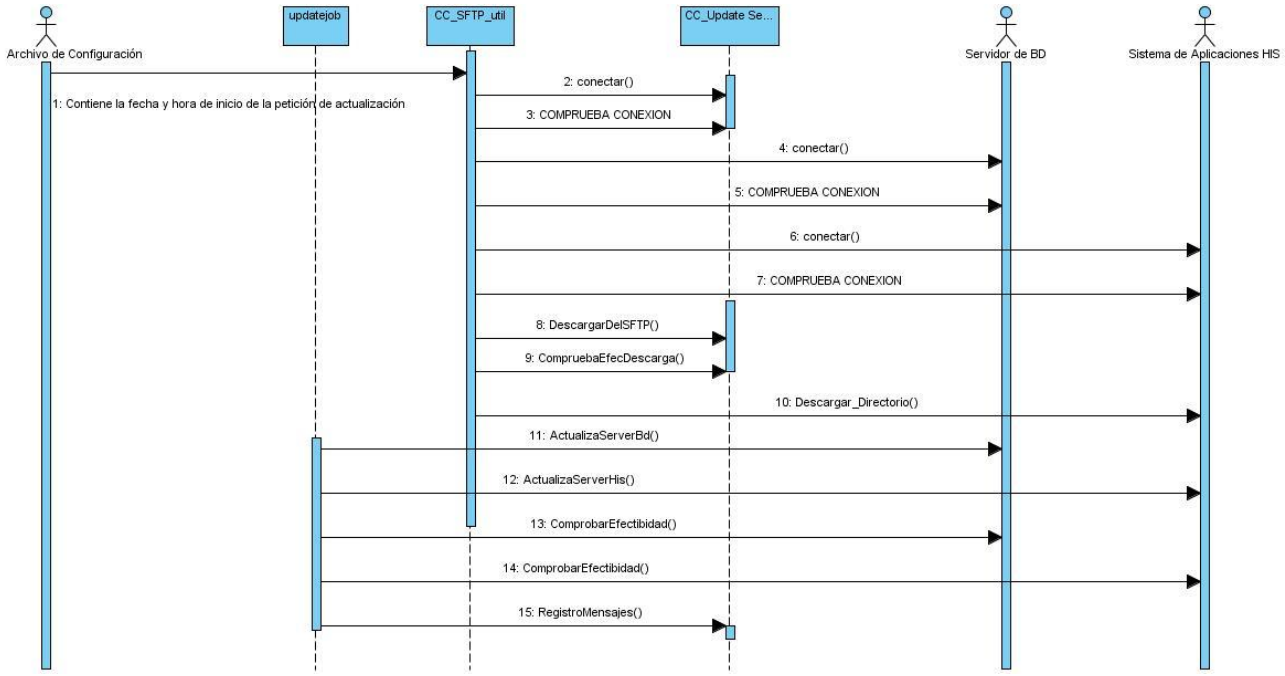

Figura 10 - Diagrama de secuencia: Gestionar actualización

3.2. Descripción de las nuevas clases u operaciones necesarias.

3.2.1. Clase: sftp_ util

Nombre: sftp_ util	
Tipo de clase controladora	
Atributo	**Tipo**
session	Session
channel	Channel
channelSftp	ChannelSftp
login	sftplogin
file	File

Para cada responsabilidad:	
Nombre:	sftp_util()
Descripción:	constructor
Nombre:	conectar()
Descripción:	Realiza las conexiones necesarias
Nombre:	isConected()
Descripción:	Verifica si está hecha la conexión
Nombre:	CopiarArchivoalSftp(String archivo_a_copiar, List directorios, String name)
Descripción:	Copia los archivos al servidor SFTP
Nombre:	Listar_Directorio(string directorio)
Descripción:	Lista los directorios en un servidor remoto.
Nombre:	DescargardelSftp(Stringdirectorio_a_descargar, Stringarchivo_a_descargar)
Descripción:	Se encarga de descargar la actualización del servidor SFTP.
Nombre:	Descargar_Directorios(String path, String local)
Descripción:	Se encarga de descargar los directorios necesarios
Nombre:	Desconectar()
Descripción:	Desconecta y devuelve si esta desconectado o no.
Nombre:	EliminarArchivodelServidor(Listpaths)
Descripción:	Devuelve si eliminó algún archivo del servidor o no.
Nombre:	CopiarDirectorio(File local, String path)
Descripción:	Copia el directorio
Nombre:	EnviarDirectorio(directorio_utilcurrent)
Descripción:	Envía el directorio a un servidor remoto
Nombre:	EnviarDirectorioaRemoto(directorio_utildirectory)
Descripción:	Envía el directorio al remoto
Nombre:	EnaviarArchivoaRemoto(File localFile, String remotePath)
Descripción:	Envía el archivo a un servidor remoto
Nombre:	Eliminar_Directorio(stringpath)

Descripción:	Eliminar el directorio totalmente
Nombre:	Eliminar_Directorio_Vacio(Stringpath)
Descripción:	Elimina los directorios vacios
Nombre:	Vaciar_Directorio (stringpath)
Descripción:	Vacía los directorios

3.1. Tabla: sftp_ util

3.2.2. Clase: servicios_ actualizador

Nombre: servicios_ actualizador	
Tipo de clase controladora	
Atributo	**Tipo**
UPDATESERVICE_WSDL_LOCATION	URL
UPDATESERVICE_EXCEPTION	WebServicesException
UPDATESERVICE_QNAME	Qname
Para cada responsabilidad:	
Nombre:	Updateservices()
Descripción:	constructor

3.2. Tabla: servicios_ actualizador

3.2.3. Clase: trabajador_actualizador

Nombre: trabajador_actualizador	
Tipo de clase (controladora)	
Atributo	**Tipo**
server	his_server

descriptor	Descriptor
respuesta	DescriptorActualizaciones
smb	smb_util
objeto_sftp_util	sftp_util
tem_dir_descargas_mod_y_core	String
tem_dir_descargas_database	String
tem_dir_backup	String
max_downloads	int
objeto_threads_pool	CompletionService<String>
url_service_location	String
objeto_db_server	db_server
server_home	String
objeto_jmx_util	jmx_util
ejecutados	List<String>
objeto_db_util	db_util
Para cada responsabilidad:	
Nombre:	updatejob()
Descripción:	constructor
Nombre:	Init(JobExecutionContextjobExecutionContext)
Descripción:	Inicializa las variables de entorno
Nombre:	execute(JobExecutionContextjobExecutionContext)
Descripción:	Inicia la actualización una vez disparada la tarea programada
Nombre:	CrearDirectoriosTemporales()
Descripción:	Crea directorios temporales
Nombre:	RealizarPeticionActualizacion()
Descripción:	Realiza la petición de actualización

Nombre:	RealizarActualizacion()
Descripción:	Inicia la actualización de forma general
Nombre:	ActualizarServerBD()
Descripción:	Actualiza el servidor de BD
Nombre:	BackupBD()
Descripción:	Lleva la base dato a su estado anterior
Nombre:	ActualizarServerHis()
Descripción:	Actualiza el servidor de aplicaciones
Nombre:	RealizarCopia()
Descripción:	Realiza la copia correspondiente a la actualización del sistema
Nombre:	LeerPath(File path)
Descripción:	Lee un fichero de texto
Nombre:	Descompactar()
Descripción:	Descompacta el paquete de actualización
Nombre:	cargar_hisdescriptor()
Descripción:	Carga el descriptor del HIS
Nombre:	HacerBackUP()
Descripción:	Hace backup
Nombre:	RestaurarBackup()
Descripción:	Restaura backup.
Nombre:	cantidad_actualizaciones()
Descripción:	Devuelve la cantidad de actualizaciones de una respuesta.

3.3. Tabla: trabajador_actualizador

3.3. Modelo de datos

Un modelo de datos permite describir los datos contenidos en tablas, la relaciones entre estas, la semántica asociada a los datos, es en general una descripción de la estructura de la base de datos con lo cual se facilita la comprensión de la solución.

3.3.1. Diagrama de la Entidad-Relación

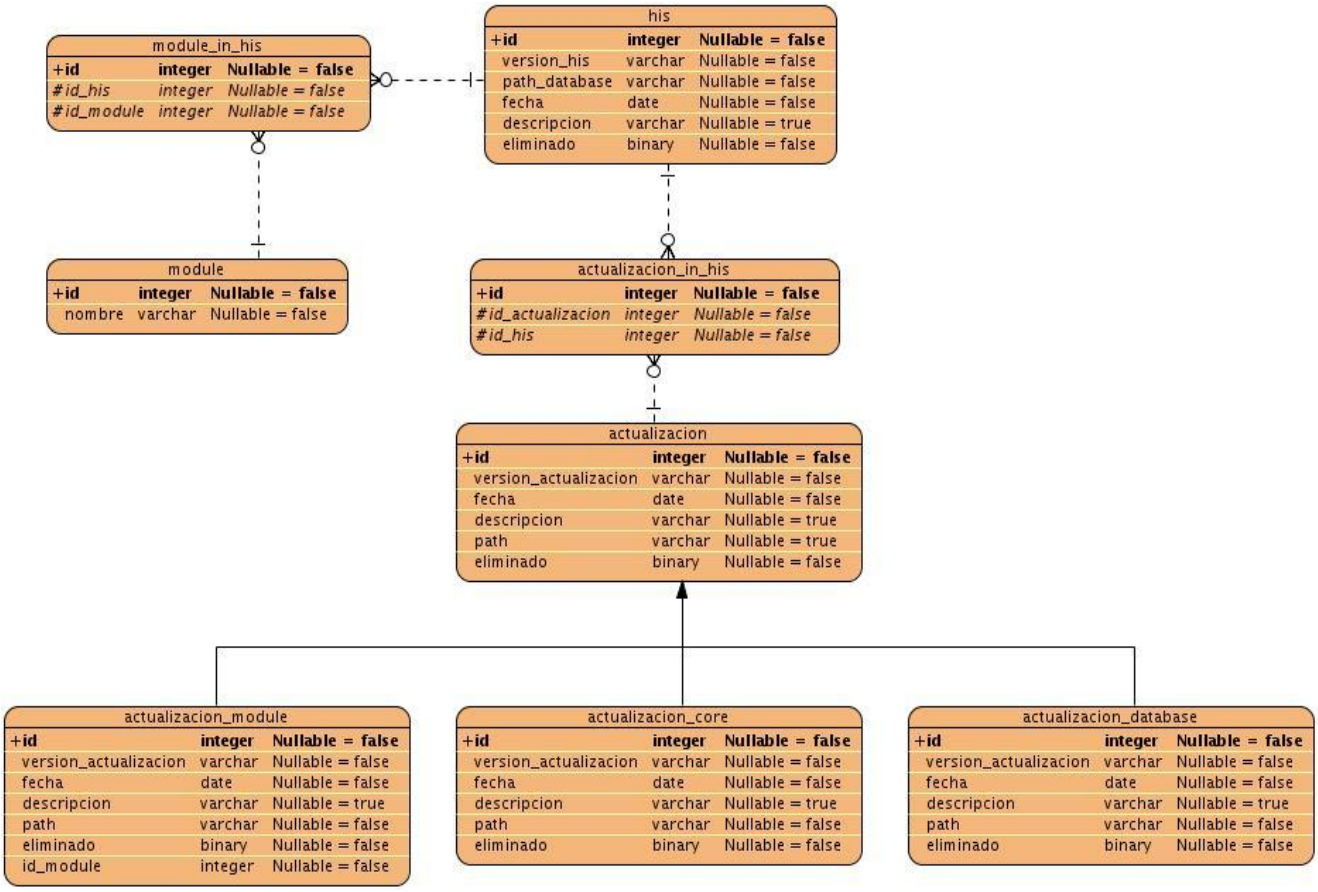

Figura 11 - Diagrama de la Entidad-Relación Infraestructura de actualización.

3.3.2. Descripción de las tablas

3.3.2.1. Tabla: actualizacion

Nombre: actualizacion		
Descripción: Tabla que contiene los datos de referentes a los paquetes de actualización.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla.
version_actualizacion	varchar	Versión correspondiente al paquete de actualización.
fecha	date	Fecha de creación del paquete actualizador.
descripcion	varchar	Breve descripción del paquete actualizador.
eliminado	boolean	Permite verificar si el paquete actualizador fue eliminado.
path	varchar	Describe el camino donde se encuentra el paquete de actualización en el servidor de actualizaciones.

3.4. Tabla: actualizacion

3.3.2.2. Tabla: líneabase_actualizacion

Nombre: líneabase_actualización		
Descripción: Tabla que contiene los datos de referentes al core de la actualización		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla.
version_actualizacion	varchar	Versión correspondiente al paquete de actualización.

fecha	date	Fecha de creación del paquete actualizador.
descripcion	varchar	Breve descripción del paquete actualizador.
eliminado	boolean	Permite verificar si el paquete actualizador fue eliminado.
path	varchar	Describe el camino donde se encuentra el paquete de actualización en el servidor de actualizaciones.

3.5. Tabla: líneabase_actualizacion

3.3.2.3. Tabla: modulo_actualizacion

Nombre: modulo_actualizacion		
Descripción: Tabla que contiene los datos de referentes a los módulos de la actualización.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla.
id_modulo	Integer	Representa el modulo para el cual fue creada la actualización.
version_actualizacion	varchar	Versión correspondiente al paquete de actualización.
fecha	date	Fecha de creación del paquete actualizador.
descripcion	varchar	Breve descripción del paquete actualizador.
eliminado	boolean	Permite verificar si el paquete actualizador fue eliminado.
path	varchar	Describe el camino donde se encuentra el paquete de actualización en el servidor de actualizaciones.

3.6. Tabla: modulo_actualizacion

3.3.2.4. Tabla: actualización_HIS

Nombre: actualización_HIS		
Descripción: Tabla que representa la relación de mucho a mucho entre la tabla actualizacion y la tabla his.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla.
id_his	integer	Representa una llave foránea de la tabla his que no es más que un identificador único.
id_actualizacion	integer	Representa una llave foránea de la tabla actualizacion que no es más que un identificador único.

3.7. Tabla: actualización_HIS

3.3.2.5. Tabla: basedatos_actualización

Nombre: basedatos_actualización		
Descripción: Tabla que contiene todos los datos de las base de datos que componen el sistema.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla.
version_actualizacion		Versión correspondiente al paquete de actualización.
path	varchar	Representa el camino donde se encuentra el paquete de actualización en el servidor de base de datos
description	varchar	Breve descripción del paquete actualizador.
fecha	date	Fecha de creación del paquete actualizador.

eliminado	boolean	Permite verificar si el paquete actualizador fue eliminado.

3.8. Tabla: basedatos_actualización

3.3.2.6. Tabla: his

Nombre: his		
Descripción: Tabla que contiene los datos referentes a las versiones del sistema.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla.
path_database	integer	Llave foránea de la tabla database que representa la base de datos.
fecha	date	Fecha de creación de la versión del HIS a que hace referencia
descripcion	varchar	Breve descripción del porque de la nueva versión del HIS
version_his	varchar	Versión del sistema la cual es única para cada sistema.
eliminado	boolean	Permite verificar si el his fue eliminado.

3.9. Tabla: his

3.3.2.7. Tabla: modulo

Nombre: modulo
Descripción: Tabla que contiene los datos de los diferentes módulos que contiene el HIS

Atributo	Tipo	Descripción
id	integer	Identificador de la tabla.
valor	varchar	Nombre del módulo.

3.10. Tabla: modulo

3.3.2.8. Tabla: modulo_his

Nombre: modulo_his		
Descripción: Tabla generada de la relación de muchos a muchos existente entre la tabla modulo y his.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla.
id_his	integer	Llave foránea que representa la tabla his.
id_mod	integer	Llave foránea que representa la tabla modulo.

3.11. Tabla: modulo_his

3.4. Breve valoración de las Técnicas de validación.

Las técnicas de validación se realizan a cualquier aplicación en desarrollo con el objetivo de corregir errores que pueden ser aprovechados por agentes externos que deseen atacar el sistema en algún momento o que simplemente pueden interferir en el correcto desenvolvimiento de la aplicación. Para facilitar este tipo de trabajo se propone el uso de las expresiones regulares, que es un mecanismo de representación de lenguajes. Además constituyen un buen procedimiento para realizar maniobras sobre cadenas de texto, que busca y/o substituye una cadena de texto por otra para ello las expresiones regulares permiten usar caracteres normales, símbolos especiales que podrán ser identificados en el contexto donde se usen y así validar si la expresión es correcta o no.

3.5. Vista de Implementación.

Los diagramas de componentes muestran una vista de la implementación pues son utilizados con el objetivo de representar los aspectos de orden físico de un sistema, dichos diagramas poseen un grupo de elementos físicos que residen en un nodo, los cuales pueden ser: bibliotecas, documentos, archivos o ejecutables.

3.5.1. Diagrama de componentes.

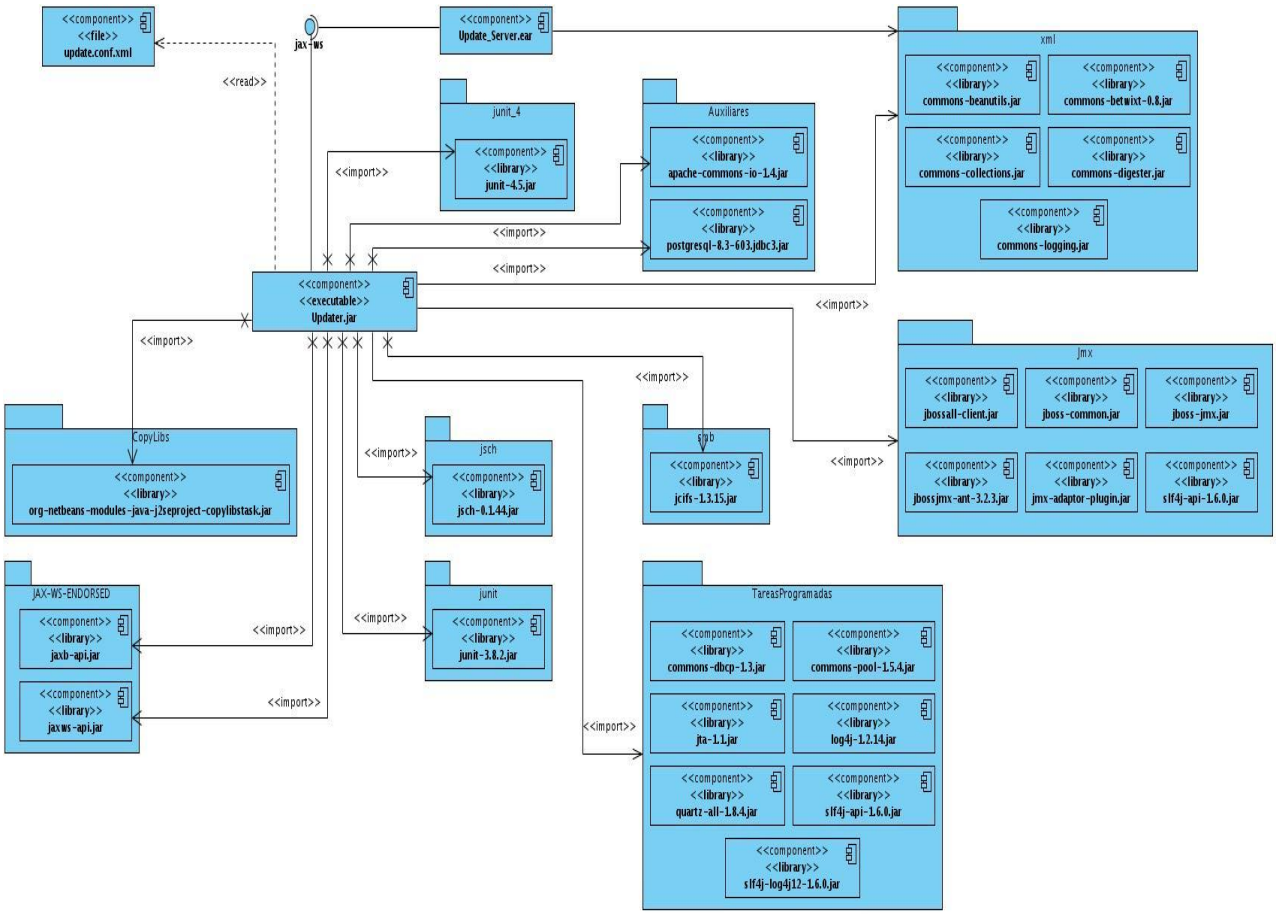

Figura 12 - Diagrama de Componentes Infraestructura de Actualización

En el capítulo se observó la descripción y análisis de la solución propuesta en el cual se pudo apreciar: los diagramas de secuencia, el diagrama entidad relación y la descripción de las tablas que lo componen, el diagrama de componentes que muestran una vista de la implementación.

Capítulo 4: Modelo de Pruebas

En el presente capítulo se expone el modelo de pruebas, con el cual se define una forma de comprobar la calidad de la solución planteada para lo que se propone el uso de la prueba de caja negra. El flujo de pruebas es muy importante en la Ingeniería de Software ya que es una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, posteriormente una evaluación es hecha de algún aspecto del sistema o componente.

Existen varias normas que sirven como objetivos de las pruebas las cuales han sido definidas por autores como Glenford Myers (28), entre las que se destacan:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

El proceso unificado de desarrollo RUP por el cual se ha guidado la construcción de esta solución propone que en cada una de las fases las pruebas se comporten de la siguiente forma:

- Inicio: el desarrollo del prototipo exploratorio de demostración; no requiere la elaboración de pruebas.
- Elaboración: probar los componentes ejecutables que se han implementado y que deben corresponderse con la arquitectura básica de la aplicación.
- Construcción: desarrollar los casos de prueba y procedimientos de prueba para hacerlos.
- Transición: el producto en su entorno de operación por lo que es probado por usuarios reales.

La prueba de software es un elemento crítico para garantizar la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación, la prueba se enfoca principalmente en la evaluación y determinación de la calidad del producto.

4.1. Prueba de caja negra

La caja negra, permite estudiar las entradas, las salidas o respuestas que se producen en un sistema sin tener en cuenta su funcionamiento interno. Permite la validación y verificación de la interfaz de usuario, donde primeramente se revisa el modelo de la interfaz para garantizar que se ajusta a los requisitos y otros elementos que puedan haber surgido durante el análisis, se tienen en cuenta los aspectos específicos de la aplicación en su interacción con los usuarios. Una de las ventajas que brinda el proceso de las pruebas es el valor agregado que le añaden al producto sobre la facilidad de uso del sistema. El objetivo fundamental de este tipo de prueba es revisar las funcionalidades del sistema, se realiza una revisión completa en busca de errores que no fueron detectados durante la prueba de caja blanca.

La caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Durante esta prueba se examinan algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca.

Algunos autores como Pressman y Beizer coinciden en que estas pruebas permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.

- Errores de rendimiento.
- Errores de inicialización y terminación.

En la preparación de los casos de prueba se necesita conocer un grupo de datos que determinaran como se ejecutara cada uno de los casos y a través de los cuales el sistema se pueda ejecutar en todas sus variantes: datos válidos o no válidos si lo que se desea probar es un error o una funcionalidad. Por lo que estos datos se determinan atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa se ejecute correctamente. Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

1. Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
2. Técnica del Análisis de Valores Límites: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
3. Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

La técnica que se ha definido para realizar estas pruebas es la de partición de equivalencia, la cual reside en dividir el dominio de entrada de una aplicación en clases de datos, derivados a partir de los mismos los casos de prueba, donde cada una de estas clases de equivalencias representa un grupo de estados válidos e inválidos para las condiciones de entrada.

4.2. Técnica de la Partición de Equivalencia

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

- Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
- Si una entrada es booleana, hay 2 clases: si o no.

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Para definir las clases de equivalencia hace falta tener en cuenta un conjunto de reglas:

- Si una condición de entrada especifica un rango, entonces se confeccionan una clase de equivalencia válida y 2 inválidas.
- Si una condición de entrada especifica la cantidad de valores, identificar una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica un conjunto de valores de entrada y existen razones para creer que el programa trata en forma diferente a cada uno de ellos, identificar una clase válida para cada uno de ellos y una clase inválida.
- Si una condición de entrada especifica una situación de tipo “debe ser”, identificar una clase válida y una inválida.
- Si existe una razón para creer que el programa no trata de forma idéntica ciertos elementos pertenecientes a una clase, dividirla en clases de equivalencia menores.

Luego de tener las clases válidas e inválidas definidas, se procede a definir los casos de pruebas, pero para ello antes se debe haber asignado un identificador único a cada clase de equivalencia. Luego entonces se pueden definir los casos teniendo en cuenta lo siguiente:

- Escribir un nuevo caso de cubra tantas clases de equivalencia válidas no cubiertas como sea posible hasta que todas las clases de equivalencia hayan sido cubiertas por casos de prueba.

- Escribir un nuevo caso de prueba que cubra una y solo una clase de equivalencia inválida hasta que todas las clases de equivalencias inválidas hayan sido cubiertas por casos de pruebas.

Con la aplicación de esa técnica se obtiene un conjunto de pruebas que reduce el númerode casos de pruebas y nos dicen algo sobre la presencia o ausencia de errores.

4.3. Descripción de los casos de prueba

4.2.1. Solicitar actualización

Escenarios del solicitar actualizaciones	**Descripción de la funcionalidad**	**Flujo Central**
Sección 1: Solicitar actualización.		
EC 1: Solicitar la actualización	La aplicación actualizadora realiza una petición de actualización al servidor de actualizaciones, satisfactoriamente.	La aplicación actualizadora realiza una petición de actualización dirigida al servidor de actualizaciones al que envía la descripción del HIS que será actualizado y la licencia del HIS. El servidor de actualizaciones recibe la petición. El servidor de actualizaciones registra la petición.
Sección 2: Comprobar validez.		
EC 2: Comprobar la validez	Servidor de actualizaciones comprueba la validez licencia del HIS, satisfactoriamente.	El servidor de actualizaciones recibe el archivo licencia del HIS. El servidor de actualizaciones comprueba la valides del la licencia del HIS.
Sección 3: Comprobar disponibilidad de actualización.		
EC 3: Comprobar disponibilidad de actualización	Servidor de actualizaciones comprueba si existen actualizaciones disponibles, satisfactoriamente.	El servidor de actualizaciones recibe el archivo la descripción del HIS. El servidor de actualizaciones comprueba si existen actualizaciones disponibles.
Sección 4: Enviar mensaje de negación de servicios.		
EC 4: Enviar mensaje de	La licencia del HIS no es válida por lo	La aplicación actualizadora realiza una petición de actualización dirigida

negación de servicios	que el servidor de actualizaciones envía mensaje de negación de servicios satisfactoriamente.	al servidor de actualizaciones al que envía la descripción del HIS que será actualizado y la licencia del HIS. El servidor de actualizaciones recibe la petición. El servidor de actualizaciones registra la petición. El servidor de actualizaciones recibe el archivo licencia del HIS. El servidor de actualizaciones comprueba la valides del la licencia del HIS. El archivo licencia no es válido. El servidor de actualizaciones envía un mensaje de negación de los servicios de actualización. El servidor de actualizaciones registra el mensaje.
Sección 5: Enviar mensaje de negación de actualización no disponible.		
EC 5: Enviar mensaje de negación de actualización no disponible	Servidor de actualizaciones envía mensaje de negación de actualizaciones disponible para la petición recibida, satisfactoriamente.	La aplicación actualizadora realiza una petición de actualización dirigida al servidor de actualizaciones al que envía la descripción del HIS que será actualizado y la licencia del HIS. El servidor de actualizaciones recibe la petición. El servidor de actualizaciones registra la petición. El servidor de actualizaciones comprueba la validez de la licencia El servidor de actualizaciones recibe el archivo la descripción del HIS. El servidor de actualizaciones comprueba si existen actualizaciones disponibles. No existen actualizaciones disponibles para el sistema que ha solicitado la actualización. El servidor de actualizaciones envía mensaje de negación de actualizaciones disponible para la petición recibida.

Id del escenario	EC 1	EC 2	EC 3	EC 4	EC 5
Escenario	Solicitar la actualización	Comprobar la validez	Comprobar disponibilidad de actualización	Enviar mensaje de negación de servicios	Enviar mensaje de negación de actualización no disponible
Respuesta del Sistema	La aplicación actualizadora realiza una petición de actualización al servidor de actualizaciones.	Servidor de actualizaciones comprueba la validez licencia del HIS, satisfactoriamente.	Servidor de actualizaciones comprueba si existen actualizaciones disponibles.	La licencia del HIS no es válida por lo que el servidor de actualizaciones envía mensaje de negación de servicios.	Servidor de actualizaciones envía mensaje de negación de actualizaciones disponible para la petición recibida.
Resultado de la Prueba					

4.2.2. Gestionar actualización

Escenarios del Gestionar actualizaciones	**Descripción de la funcionalidad**	**Flujo Central**
Sección 1: Establecer conexión segura		
EC 6: Establecer conexión segura.	El sistema estable una conexión segura con el servidor SFTP, con la base de datos del HIS y con el HIS, satisfactoriamente.	El sistema establece una conexión segura con el servidor SFTP utiliza los datos siguientes para establecer la conexión: • Host • Puerto • Usuario • Contraseña El sistema establece conexión segura a la base de datos del

		sistema HIS, con los datos siguientes para establecer la conexión. • Host • Puerto • Usuario • Contraseña • Nombre de la base de datos El sistema establece conexión segura al HIS. Utiliza los datos siguientes para establecer la conexión: • Host • Puerto • Usuario • Contraseña
Sección 2: Descargar paquetes de actualización		
EC 7: Descargar paquetes de actualización.	El sistema descarga los paquetes de actualización, satisfactoriamente.	El sistema descarga los paquetes de actualización desde el servidor de actualizaciones a través de la conexión segura.
Sección 3: Comprobar efectividad de la descarga		
EC 8: Comprobar efectividad de la descarga.	El sistema comprueba la efectividad de la descarga, satisfactoriamente.	El sistema comprueba la efectividad de la descarga realizada desde el servidor de actualizaciones hacia la aplicación actualizadora.
Sección 4: Realizar copia de seguridad		
EC 9: Realizar copia de seguridad.	El sistema realiza copia de seguridad, satisfactoriamente.	El sistema realiza una salva de la aplicación en uso. El sistema guarda la salva en un archivo temporal destinado para el proceso.
Sección 5: Actualizar servidor de base de datos		
EC 10: Actualizar servidor de	El sistema actualiza servidor de base	El sistema copia los paquetes correspondientes a la base de

base de datos.	datos, satisfactoriamente.	dato en el directorio del servidor de base datos. El sistema ejecuta redespliegue para que los cambios efectuados tengan efecto en el sistema.
Sección 6: Actualizar servidor de aplicaciones		
EC 11: Actualizar servidor de aplicaciones.	El sistema actualiza servidor de aplicaciones, satisfactoriamente.	El sistema copia los paquetes de actualización del directorio temporal donde estaban guardados hacia el directorio de la aplicación. El sistema ejecuta redespliegue para que los cambios efectuados tengan efecto en el sistema.
Sección 7: Enviar mensaje de efectividad de actualización		
EC 12: Enviar mensaje de efectividad de actualización.	El sistema envía mensaje de efectividad, satisfactoriamente.	El sistema envía un mensaje que comunica la efectividad de la actualización.
Sección 8: Los datos de la conexión no fueron correctos		
EC 13: Los datos de la conexión no fueron correctos.	El sistema envía mensaje de datos de la conexión incorrectos, satisfactoriamente.	El sistema establece una conexión segura con el servidor SFTP utiliza los datos siguientes para establecer la conexión: • Host • Puerto • Usuario • Contraseña El sistema establece conexión segura a la base de datos del sistema HIS, con los datos siguientes para establecer la conexión. • Host • Puerto • Usuario • Contraseña • Nombre de la base de datos

		El sistema establece conexión segura al HIS. Utiliza los datos siguientes para establecer la conexión: • Host • Puerto• Usuario • Contraseña Si alguno de estos datos no es correcto el sistema suspende el proceso de actualización
Sección 9: Comprobar efectividad de actualización y restablecer versión anterior DB		
EC 14: Comprobar efectividad de actualización y restablecer versión anterior DB.	El sistema restablece versión anterior de la base de datos, satisfactoriamente.	El proceso de actualización no fue efectivo el sistema restablece la versión anterior. El sistema envía mensaje de fallo de actualización. El sistema registra el proceso.
Sección 10: Comprobar efectividad de actualización y restablecer versión anterior del sistema		
EC 15: Comprobar efectividad de actualización y restablecer versión anterior del sistema.	El sistema restable versión anterior del sistema, satisfactoriamente.	Si la actualización no fue efectiva se restablece versión anterior. El sistema restablece la versión anterior. El sistema envía mensaje de fallo de actualización. El sistema registra el proceso.

Id del escenario	EC1	EC2	EC3	EC4	EC5	EC6	EC7	EC8	EC9	EC10

Escenario	Establecer conexión segura.	Descargar paquetes de actualización.	Comprobar efectividad de la descarga.	Realizar copia de seguridad.	Actualizar servidor de base de datos.	El sistema actualiza servidor de aplicaciones	El sistema envía mensaje de efectividad, satisfactoriamente.	El sistema envía mensaje de datos de la conexión incorrectos, satisfactoriamente.	El sistema restablece versión anterior de la base de datos, satisfactoriamente.	El sistema restable versión anterior del sistema, satisfactoriamente.
Respuesta del Sistema	El sistema estable una conexión segura con el servidor SFTP, con la base de datos del HIS y con el HIS.	El sistema descarga los paquetes de actualización.	El sistema comprueba la efectividad de la descarga.	El sistema realiza copia de seguridad.	El sistema actualiza servidor de base datos.	El sistema actualiza servidor de aplicaciones.	El sistema envía mensaje de efectividad.	El sistema envía mensaje de datos de la conexión incorrectos.	El sistema restablece versión anterior de la base de datos.	El sistema restable versión anterior del sistema.
Resultado de la Prueba										

En este capítulo se observó la importancia del flujo de pruebas en la Ingeniería de Software, destacándose los beneficios que reporta este proceder al desarrollo de cualquier sistema, ya que las pruebas se enfocan principalmente en la evaluación y determinación de la calidad del producto. Finalmente se seccionó la Prueba de Caja Negra con la cual se pueden estudiar las entradas, las salidas o respuestas que se producen en un sistema sin tener en cuenta su funcionamiento interno. Además se mencionaron las diferentes técnicas existentes dentro de la misma, escogiéndose para ser utilizada la Técnica de la Partición Equivalente; la cual divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software, para lo cual se construyeron casos de prueba basados en las clases de equivalencia, los cuales fueron expuestos anteriormente.

Conclusiones

Con el desarrollo del presente trabajo y el cumplimiento de las tareas propuestas se plantean las siguientes conclusiones:

La investigación de las metodologías, plataformas, tecnologías, librerías y herramientas permitieron el desarrollo de un sistema robusto, estable y flexible que garantiza la seguridad y el correcto desempeño de la actualización automática.

El estudio de los diferentes sistemas demostró que ninguno cumple con las condiciones y características necesarias, lo que evidencia la necesidad de desarrollar una infraestructura robusta que garantice el proceso de actualización automática en el Sistema de Información Hospitalaria alas HIS.

La arquitectura propuesta para la solución garantiza una mejor ejecución de los procesos de comunicación pues definió la organización del sistema de software, su comportamiento y las colaboraciones entre sus elementos.

La realización del diseño, en correspondencia con la arquitectura, facilitó la implementación de los procesos correspondientes a la Infraestructura de Actualización, obteniéndose satisfactoriamente los artefactos correspondientes a los flujos de trabajo de “Implementación” y “Pruebas”.

Se implementó el sistema Infraestructura de Actualización para el Sistema de Información Hospitalaria alas HIS acorde con lo establecido en la Especificación de Requisitos realizada donde se aplicaron las pautas de diseño definidas.

Referencias Bibliográficas

Trabajos citados

1. **ALBET.** alas HIS | Portal de ALBET Ingeniería y Sistemas . [En línea] [Citado el: 10 de 12 de 2010.] http://www.albet.cu/lineas/salud-y-ciencias/alas-his.

2. *A Characterization Framework for Software Deployment Technologies.* **Carzaniga, Antonio y Fuggettaz, Alfonso y otros.** University of Colorado : Department of Computer Science, April 1998, Vols. Technical Report CU-CS-857-98.

3. **Jansen, Slinger y Ballintijn, Gerco.** A Process Model and Typology for Software Product Updaters. *citeseerx.* [En línea] 2007-2010 The Pennsylvania State University,Developed at and hosted by The College of Information Sciences and Technology at Penn State, 2005 . [Citado el: 1 de junio de 2011.] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.5074.

4. EcuRed. *EcuRed.* [En línea] Cuba. [Citado el: 28 de 2 de 2011.] http://www.ecured.cu/index.php/proveedor.

5. EcuRed. *EcuRed.* [En línea] Cuba. [Citado el: 28 de 2 de 2011.] http://www.ecured.cu/index.php/cliente.

6. **Corporation, Symantec.** Symantec. *Symantec.* [En línea] Symantec Corporation, 1995 - 2011. [Citado el: 17 de marzo de 2011.] http://www.symantec.com/index.jsp.

7. **MARTÍNEZ CABRERA, JORGE ERNESTO y MILIÁN RODRIGUEZ, RAISEL.** *Mecanismo de actualización para el Sistema de Gestión Penitenciaria Venezolano. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* Habana, Cuba : UCI, JULIO DE 2007.

8. **Céspedes, Paúl y Jesús, Armando.** Manuales SIGHO, ACTUALIZADOR version del módulo 3.3.0. *SIGHO, Sistema de Informacion para la Gerencia Hospitalaria.* [En línea] [Citado el: 17 de marzo de 2011.] http://sigho.saludtlax.gob.mx/?p=5.

9. **Ladrón de Guevara Chala, Yisel Taimi y Miranda Silva, Rubén.** *Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas Propuesta de infraestructura de software para la gestión de la Historia Clínica Electrónica centralizada.* Habana : UCI, Julio de 2010.

10. **Díaz Rodríguez, Jorge.** *Implementación de los procesos del movimiento hospitalario del Sistema de Información Hospitalaria alas HIS.* La Habana : UCI, junio de 2010.

11. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** Racional Unified Process. *Google libros.* [En línea] [Citado el: 2011 de 2 de 28.] http://books.google.es/books?hl=es&lr=&id=RYCMx6o47pMC&oi=fnd&pg=PR13&dq=Racional+Unified+Process&ots=h0asAUO6Ud&sig=LuuvBDdIYenr5_3hzhDIPZj6CUk#v=onepage&q&f=false.

12. Unified Modeling Languaje. *Unified Modeling Languaje.* [En línea] Object Management Group, Copyright © 1997-2011. [Citado el: 21 de 02 de 2011.] http://www.uml.org/.

13. Java. *Java.* [En línea] ORACLE. [Citado el: 04 de 02 de 2011.] http://www.java.com..

14. **javaHispano, Asociación.** javaHispano. *javaHispano.* [En línea] Asociación javaHispano, 2002- 2007. [Citado el: 04 de 02 de 2011.] http://www.javahispano.org/.

15. seamframework. *seamframework.* [En línea] Copyright 2009, Red Hat Middleware. [Citado el: 20 de marzo de 2011.] http://seamframework.org/.

16. HIBERNATE, Jboss Community. *HIBERNATE.* [En línea] redhat. [Citado el: 29 de 6 de 2011.] http://www.hibernate.org/.

17. ORACLE. *ORACLE.* [En línea] ORACLE. [Citado el: 29 de 6 de 2011.] http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html.

18. java.net. *java.net.* [En línea] Java, 28 de 04 de 2007. [Citado el: 04 de 02 de 2011.] http://wiki.java.net/bin/view/Javawsxml/Jax-wsSI.

19. **Angel Alvarez, Miguel.** desarrolloweb.com. *desarrolloweb.com.* [En línea] Licencia copyleft. [Citado el: 29 de 6 de 2011.] http://www.desarrolloweb.com/descargas/descargar.php?descarga=2754.

20. Open SSH. *Open SSH.* [En línea] OpenBSD, 1999-2009. [Citado el: 05 de 02 de 2011.] http://www.openbsd.org/cgi-bin/man.cgi?query=sftp&sektion=1.

21. Visual Paradigm. *Visual Paradigm.* [En línea] Visual Paradigm International. [Citado el: 04 de 02 de 2011.] http://www.visual-paradigm.com..

22. eclipse. *eclipse.* [En línea] Copyright © 2011 The Eclipse Foundation. All Rights Reserved. [Citado el: 29 de 6 de 2011.] http://www.eclipse.org/ganymede/.

23. PgAdmin PostgreSQL Tools. *PgAdmin PostgreSQL Tools.* [En línea] [Citado el: 04 de 02 de 2011.] http://www.pgadmin.org.

24. Jboss Community. Jboss Application server. *Jboss Community. Jboss Application server.* [En línea] Red Hat. . [Citado el: 04 de 02 de 2011.] http://www.jboss.org/jbossas/downloads.html..

25. Netbeans. . *Netbeans. .* [En línea] Oracle Corporation and/or its affiliates., 2011. [Citado el: 04 de 02 de 2011.] http://netbeans.org..

26. Postgresql. . *Postgresql.* [En línea] PostgreSQL Global Development Group, 1996- 2011. [Citado el: 04 de 02 de 2011.] http://www.postgresql.org..

27. OpenSSH. *OpenSSH.* [En línea] This site Copyright © 1999-2009 OpenBSD. [Citado el: 29 de 6 de 2011.] http://www.openssh.com/.

28. Glenford Myers ; The art of software testing; Citado por 2759. *Glenford Myers ; The art of software testing; Citado por 2759.* [En línea] [Citado el: 29 de 6 de 2011.] http://www.carlosfau.com.ar/nqi/nqifiles/The%20Art%20of%20Software%20Testing%20-%20Second%20Edition.pdf.

Bibliografía

1. **ALBET.** alas HIS | Portal de ALBET Ingeniería y Sistemas . [Online] [Cited: 12 10, 2010.] http://www.albet.cu/lineas/salud-y-ciencias/alas-his.

2. *A Characterization Framework for Software Deployment Technologies.* **Carzaniga, Antonio and Fuggettaz, Alfonso y otros.** University of Colorado : Department of Computer Science, April 1998, Vols. Technical Report CU-CS-857-98.

3. **Jansen, Slinger and Ballintijn, Gerco.** A Process Model and Typology for Software Product Updaters. *citeseerx.* [Online] 2007-2010 The Pennsylvania State University,Developed at and hosted by The College of Information Sciences and Technology at Penn State, 2005 . [Cited: junio 1, 2011.] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.5074.

4. EcuRed. *EcuRed.* [Online] Cuba. [Cited: 2 28, 2011.] http://www.ecured.cu/index.php/proveedor.

5. EcuRed. *EcuRed.* [Online] Cuba. [Cited: 2 28, 2011.] http://www.ecured.cu/index.php/cliente.

6. **Corporation, Symantec.** Symantec. *Symantec.* [Online] Symantec Corporation, 1995 - 2011. [Cited: marzo 17, 2011.] http://www.symantec.com/index.jsp.

7. **MARTÍNEZ CABRERA, JORGE ERNESTO and MILIÁN RODRIGUEZ, RAISEL.** *Mecanismo de actualización para el Sistema de Gestión Penitenciaria Venezolano. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* Habana, Cuba : UCI, JULIO DE 2007.

8. **Céspedes, Paúl and Jesús, Armando.** Manuales SIGHO, ACTUALIZADOR version del módulo 3.3.0. *SIGHO, Sistema de Informacion para la Gerencia Hospitalaria.* [Online] [Cited: marzo 17, 2011.] http://sigho.saludtlax.gob.mx/?p=5.

9. **Ladrón de Guevara Chala, Yisel Taimi and Miranda Silva, Rubén.** *Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas Propuesta de infraestructura de software para la gestión de la Historia Clínica Electrónica centralizada.* Habana : UCI, Julio de 2010.

10. **Díaz Rodríguez, Jorge.** *Implementación de los procesos del movimiento hospitalario del Sistema de Información Hospitalaria alas HIS.* La Habana : UCI, junio de 2010.

11. **Jacobson, Ivar, Booch, Grady and Rumbaugh, James.** Racional Unified Process. *Google libros.* [Online] [Cited: 2 2011, 28.] http://books.google.es/books?hl=es&lr=&id=RYCMx6o47pMC&oi=fnd&pg=PR13&dq=Racional+Unified+Process&ots=h0asAUO6Ud&sig=LuuvBDdIYenr5_3hzhDIPZj6CUk#v=onepage&q&f=false.

12. Unified Modeling Languaje. *Unified Modeling Languaje.* [Online] Object Management Group, Copyright © 1997-2011. [Cited: 02 21, 2011.] http://www.uml.org/.

13. Java. *Java.* [Online] ORACLE. [Cited: 02 04, 2011.] http://www.java.com..

14. **javaHispano, Asociación.** javaHispano. *javaHispano.* [Online] Asociación javaHispano, 2002- 2007. [Cited: 02 04, 2011.] http://www.javahispano.org/.

15. seamframework. *seamframework.* [Online] Copyright 2009, Red Hat Middleware. [Cited: marzo 20, 2011.] http://seamframework.org/.

16. HIBERNATE, Jboss Community. *HIBERNATE.* [Online] redhat. [Cited: 6 29, 2011.] http://www.hibernate.org/.

17. ORACLE. *ORACLE.* [Online] ORACLE. [Cited: 6 29, 2011.] http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html.

18. java.net. *java.net.* [Online] Java, 04 28, 2007. [Cited: 02 04, 2011.] http://wiki.java.net/bin/view/Javawsxml/Jax-wsSI.

19. **Angel Alvarez, Miguel.** desarrolloweb.com. *desarrolloweb.com.* [Online] Licencia copyleft. [Cited: 6 29, 2011.] http://www.desarrolloweb.com/descargas/descargar.php?descarga=2754.

20. Open SSH. *Open SSH.* [Online] OpenBSD, 1999-2009. [Cited: 02 05, 2011.] http://www.openbsd.org/cgi-bin/man.cgi?query=sftp&sektion=1.

21. Visual Paradigm. *Visual Paradigm.* [Online] Visual Paradigm International. [Cited: 02 04, 2011.] http://www.visual-paradigm.com..

22. eclipse. *eclipse.* [Online] Copyright © 2011 The Eclipse Foundation. All Rights Reserved. [Cited: 6 29, 2011.] http://www.eclipse.org/ganymede/.

23. PgAdmin PostgreSQL Tools. *PgAdmin PostgreSQL Tools.* [Online] [Cited: 02 04, 2011.] http://www.pgadmin.org.

24. Jboss Community. Jboss Application server. *Jboss Community. Jboss Application server.* [Online] Red Hat. . [Cited: 02 04, 2011.] http://www.jboss.org/jbossas/downloads.html..

25. Netbeans. . *Netbeans. .* [Online] Oracle Corporation and/or its affiliates., 2011. [Cited: 02 04, 2011.] http://netbeans.org..

26. Postgresql. . *Postgresql.* [Online] PostgreSQL Global Development Group, 1996- 2011. [Cited: 02 04, 2011.] http://www.postgresql.org..

27. OpenSSH. *OpenSSH.* [Online] This site Copyright © 1999-2009 OpenBSD. [Cited: 6 29, 2011.] http://www.openssh.com/.

28. Glenford Myers ; The art of software testing; Citado por 2759. *Glenford Myers ; The art of software testing; Citado por 2759.* [Online] [Cited: 6 29, 2011.] http://www.carlosfau.com.ar/nqi/nqifiles/The%20Art%20of%20Software%20Testing%20-%20Second%20Edition.pdf.

29. *Sistema de Información Hospitalaria.* **Florina, Gatica Lara, J., Fernando y Puerto, Fernández.** s.l. : Facultad de Medicina, Vol. S.l: UNAM.

30. *A process framework and typology for software product updaters.* **Jansen, Slinger, Ballintijn, Gerco y Brinkkemper, Sjaak.** Amsterdam, The Netherlands : Centre for Mathematics and Computer Science; U, Institute of Information and Computing Sciences, Vol. Volume 10.

31. **Garzón, Darwin Jiménez.** *Ingeniería de Software II.*

32. **Hat., JBoss by Red.** JBoss by Red Hat. . *JBoss by Red Hat. .* [Online] Red Hat, Inc., 2011. [Cited: 02 04, 2011.] http://www.jboss.com/products/hibernate/.

33. Oracle. *Oracle.* [Online] [Cited: 02 21, 2011.] http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html.

34. **javaHispano.** javaHispano. *javaHispano.* [Online] 2002-2007 Asociación javaHispano. [Cited: 02 04, 2011.] http://www.javahispano.org/contenidos/es/licencia_para_aplicaciones_java/..

35. **Eclipse.** EclipseCON. *EclipseCON.* [Online] The Eclipse Foundation. , 2011. [Cited: 02 04, 2011.] http://www.eclipse.org.

36. **Std, IEEE.** *IEEE Software Engineering Standard: Glossary of Software Engineering Terminology.* s.l. : EEE Computer Society Press, 1993.

37. EcuRed. *EcuRed.* [Online] [Cited: 02 28, 2011.] http://www.ecured.cu/index.php/Software.

38. **Slinger Jansen, Gerco Ballintijn, Sjaak Brinkkemper.** *A Process Model and Typology for Software Product Updaters.* Amsterdam, The Netherlands : Centre for Mathematics and Computer Science.

39. *A Cooperative Approach to Support Software Deployment Using the Software Dock.* **Hall, Richard S., D.H. y Wolf., Alexander.** University of Colorado, USA : Department of Computer Science, 1998.

40. **Dueñas, Joel Barrios.** Implementación De Servidores Con GNU/Linux. [Online] Edición Julio 2008. [Cited: 3 20, 2011.] http://arpaneting.es/wp-content/uploads/2008/07/implementacion_servidores_linux-julio-2008.pdf#page=249.

41. mastermagazine*mastermagazine*http://www.mastermagazine.info/termino/4294.php

42. **Juntao Yuan, Michael and Heute, Thomas.** *Jboss Seam Simplicity And Power Beyond Java EE.*

43. *A Characterization Framework for Software Deployment Technologies.* **Carzaniga, Antonio, et al.** University of Colorado : Department of Computer Science, April 1998, Vols. Technical Report CU-CS-857-98.