Universidad de las Ciencias Informáticas
Facultad 7

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Implementación de la Hoja de trasplante del componente Consulta Externa de Nefrología del Sistema de Información Hospitalaria alas HIS

Autora: Sirleys Bisset Despaigne

Tutora: Ing. Yanersy Díaz Colomé

Cotutora: Ing. Claudia Fuentes Escobar

La Habana, junio 2011

"Año 53 de la Revolución"

"Porque mejor es la sabiduría que las piedras preciosas, Y todo cuanto se puede desear, no es de compararse con ella…"

Proverbios 8:11

Ing. Yanersy Díaz Colomé

Graduada de Ingeniería en Ciencias Informáticas, en la UCI, en el curso 2006-2007. Profesor Instructor. Durante su trabajo como profesor ha impartido las asignaturas de Matemática I, Matemática II, Gestión de Software, Sistemas de Bases de Datos I y Sistemas de Bases de Datos II.

En la vinculación con la producción pertenece al Departamento de Sistemas Especializados en Medicina del Centro Especializado en Soluciones de Informática Médica y se desempeña como Analista del proyecto Nefrología.

Correo electrónico: ydiaz@uci.cu .

Ing. Claudia Fuentes Escobar

Instructora recién graduada en el año 2010 de Ingeniería en Ciencias Informáticas en la UCI. Profesora vinculada a la Facultad 7 y miembro del Departamento de Sistemas Especializados en Medicina del Centro Especializado en Soluciones de Informática Médica.

Correo electrónico: cfuentes@uci.cu.

AGRADECIMIENTOS

- Gracias al Espíritu Santo de Dios, por siempre estar a mi lado, dándome fuerzas para que pueda continuar, porque verdaderamente las necesitaba para poder seguir adelante.
- Gracias a mis padres María Despaigne y Oreste Bisset por haberme traído al mundo y ser la persona que soy, a mi hermano Osmani Bisset.
- A mis hermanos en Cristo, son tremenda bendición, especialmente a Moreira, Héctor, Amarilis, Livan, Odalis, Adolfo, Yaneidis, Alexei Alayo.
- A mi familia por siempre estar atenta a todos mis problemas, especialmente a mi abuela Ofelia Despaigne, mi tía Juana, mi tío Santiago, mis tíos del Cotorro Alberto, Anita, a mi primita Dunays, a mi primo Yansel, Albertico ya todos los demás por preocuparse por mí.
- A mis tutoras Claudia y Yanersy.
- A mi prima Ivet.
- A los profesores del proyecto, a Javier, Leosdan, Brito, Yanoska.
- A mis compañeros de aula y de apartamento.
- Al compañero de la Revolución Cubana Fidel Castro Ruz por realizar un proyecto tan precioso como la Universidad de las Ciencias Informáticas.
- A la Revolución Cubana por brindar la posibilidad de estudios gratis.

DEDICATORIA

- Al Espíritu Santo de Dios, por ser siempre mi pastor.
- A mis padres María Despaigne y Oreste Bisset, y a hermano Osmani Bisset.
- A mi familia por brindarme su apoyo. Especialmente mi abuelita Ofelia Despaigne
- A mis hermanos en Cristo.

Resumen

Los pacientes que padecen enfermedades renales y requieren de trasplante renal, se atienden en los hospitales donde se brinda un servicio de Consulta Externa de Nefrología, la información referente a dichos pacientes se guardan en una historia clínica de papel, lo que provoca que los datos puedan perderse o extraviarse.

En la universidad se está desarrollando el Sistema de Información Hospitalaria alas HIS que tiene varias áreas como: Traumatología y Ortopedia, Neurología, Ginecología entre otras, pero no cuenta con un servicio de Consulta Externa de Nefrología, que gestione los datos de los pacientes y los posibles donantes implicados en el proceso de trasplante renal.

El presente trabajo tiene como objetivo general Implementar la Hoja de trasplante renal del componente Consulta Externa de Nefrología del Sistema de Información Hospitalaria alas HIS. Para el diseño de la Hoja de trasplante renal se utilizó como metodología de desarrollo de software el Proceso Unificado de Desarrollo (RUP) para especificar, construir y documentar el sistema. Se hizo uso del Lenguaje Unificado de Modelado (UML) y Visual Paradigm 6.4 para la creación de los artefactos que se generan durante el ciclo de vida del software. Se utilizó el Entorno Integrado de Desarrollo Eclipse con el lenguaje Java y como servidor de aplicación Jboss.

La implementación de la Hoja de trasplante renal, permitirá una mayor eficiencia en la gestión de la información de los pacientes que son atendidos en la Consulta Externa de Nefrología y que se encuentran en el Programa de Trasplante.

PALABRAS CLAVES

Hoja de trasplante renal, donante, Consulta Externa de Nefrología.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La informatización de cada esfera de la sociedad se hace cada vez más necesaria en todo el mundo, llevarla a cabo, es una tarea que requiere de grandes esfuerzos. La salud es uno de los sectores beneficiados por este proceso. En Cuba se desarrolla un programa de informatización dirigido fundamentalmente a la salud pública, que tiene como objetivo brindar un servicio de alta calidad.

Dentro del marco de la salud pública se encuentra la Nefrología, especialidad que se define como: el estudio de la estructura y función renal, tanto en la salud como en la enfermedad, que incluye la prevención y tratamiento de las enfermedades que afectan al riñón. [1]

A la Nefrología se le asignan considerables recursos desde el triunfo de la Revolución, uno de los proyectos sociales para la salud más conocido fue la adquisición de un riñón artificial para un hospital de Ciudad de La Habana.

Más tarde, luego de la creación del Instituto de Nefrología (INEF), se funda el Centro Nacional Coordinador de Trasplante. Este centro ha permitido hasta hoy el intercambio de órganos (riñones) entre todos los territorios, la posibilidad de acceder a la diálisis y al trasplante renal.

Los pacientes que padecen enfermedades renales, que por su severidad requieren de atención especializada, se atienden en hospitales en el servicio de Consulta Externa de Nefrología. La información referente a dichos pacientes se guardan en una historia clínica en formato duro, lo que provoca que los datos puedan perderse por el deterioro de estos documentos o extraviarse en caso de que el enfermo deba de recurrir a otro centro, pues él sería el encargado de trasladar su historia clínica.

El registro de aquellos pacientes que se atienden en el Programa de Trasplante se lleva a cabo en los 48 servicios de Nefrología con los que cuenta el país. De estos servicios nueve son centros de trasplante renal y cinco incluyen laboratorios de histocompatibilidad.

Los datos de los pacientes que se atienden en cada centro se recogen diariamente. Esta información es tramitada hacia el Instituto de Nefrología una vez al mes mediante correo electrónico o vía telefónica. Dicho proceder provoca que se puedan cometer errores humanos, que no se pueda obtener la información de los pacientes en tiempo real, ni con la exactitud requerida. Lo que implica que la gestión de la lista de espera para trasplantes renales sea engorrosa, pues en un momento dado se pueden tener falsos aptos. Esto provoca en muchos casos que el órgano se pierda, ya que la vida útil de un riñón después de extraído es de 24 horas. [2]

El trasplante renal se puede realizar de donante vivo o de donante cadáver. Al receptor y a los donantes vivos se les efectúa un seguimiento previo al trasplante, donde se le realizan una serie de exámenes para determinar si ambos están aptos para ser sometidos a esta cirugía.

Dicho seguimiento se registra en la historia clínica del paciente y en las historias clínicas de sus posibles donantes (en caso de donante vivo). Las historias clínicas de los donantes se anexan a la del paciente. Esto implica que la gestión de la información sea bastante compleja y que la acumulación de documentos alcance niveles muy altos.

Para dar solución a la situación planteada anteriormente, en el curso 2007-2008 la facultad 7 de la Universidad de las Ciencias Informáticas desarrolló el sistema alas NEFRORED v1.0; que permite viabilizar la gestión de la información relacionada con los procesos de trasplante renal, específicamente la asociada con el seguimiento previo al trasplante para pacientes que están en espera de esta cirugía y el seguimiento previo al trasplante de los posibles donantes (en caso de tratarse de donante vivo).

Actualmente se está desarrollando el Sistema de Gestión Hospitalaria (alas HIS) encargado de almacenar, generar y procesar datos médico-administrativos de las distintas áreas de las instituciones hospitalarias. Una de estas áreas es la Consulta Externa, que a su vez consta de varios servicios, entre los que se destacan: Traumatología y Ortopedia, Neurología y Ginecología. Sin embargo, este sistema no cuenta con una Consulta Externa de Nefrología, donde se pueda atender a pacientes que presenten padecimientos renales; por lo que no se gestiona la información relacionada con aquellos que requieran trasplante renal, el estado de aptitud en tiempo real de los mismos, los datos de sus posibles donantes y la idoneidad de estos.

No se le puede integrar la aplicación alas NEFRORED porque las arquitecturas son incompatibles. Por lo cual se hace necesario realizar un nuevo desarrollo para el sistema alas HIS, que incluya la gestión de la información relacionada con el trasplante renal en la Consulta Externa de Nefrología. Para ello en el curso 2009 – 2010 se realizó el diseño de la Hoja de trasplante renal del componente Consulta Externa de Nefrología, pero dicho diseño no es funcional.

Dada la problemática anteriormente expuesta se plantea el siguiente **problema a resolver**:

¿Cómo hacer funcional el diseño de la Hoja de trasplante renal del componente Consulta Externa de Nefrología del Sistema de Información Hospitalaria alas HIS? Se define como **objeto de estudio** el proceso de desarrollo del componente Consulta Externa de Nefrología del Sistema de Información Hospitalaria alas HIS.

El objeto delimita el **campo de acción**: el proceso de implementación y prueba de las funcionalidades relacionadas con la Hoja de trasplante renal del componente Consulta Externa de Nefrología del Sistema de Información Hospitalaria alas HIS.

Como **objetivo** de la investigación se definió: Implementar la Hoja de trasplante renal del componente Consulta Externa de Nefrología del Sistema de Información Hospitalaria alas HIS, a partir del diseño realizado.

Para el cumplimiento del objetivo planteado se han propuesto las siguientes **tareas de la investigación**:

- Analizar el proceso de negocio relacionado con el trasplante renal en la Consulta Externa de Nefrología descrito por el analista.
- Describir la arquitectura propuesta por el Departamento de Gestión Hospitalaria.
- Realizar una valoración crítica del diseño propuesto por el analista.
- Obtener los artefactos correspondientes a los Flujos de Trabajo: "Implementación" y "Pruebas".
- Implementar la Hoja de trasplante renal del componente Consulta Externa de Nefrología, donde se aplicarán las pautas de diseño definidas y a partir de las necesidades de funcionamiento establecidas en la Especificación de Requisitos de Software.

El presente documento está estructurado en cuatro capítulos que contienen la información relacionada con la investigación realizada. Estos son:

- **Capítulo 1**. Fundamentación teórica: abordará lo referente al módulo Consulta Externa y el proceso relacionado con el trasplante renal llevado a cabo en la Consulta Externa de Nefrología. Así como las funcionalidades a implementar y la descripción de las tecnologías, metodologías y herramientas a utilizar para el desarrollo del sistema.
- **Capítulo 2**. Descripción de la arquitectura: se describe la arquitectura que soportará el desarrollo de la aplicación, definiéndose además los requerimientos no funcionales, la estrategia de integración a otros módulos, la seguridad a implementar en el sistema y la estrategia de codificación.
- **Capítulo 3**. Descripción y análisis de la solución propuesta: se centra en la valoración del diseño propuesto por el analista y la construcción de la aplicación.
- **Capítulo 4**. Prueba: refleja el método de prueba a utilizar y se describen los casos de uso correspondientes.

CAPÍTULO 1: FUNDAMENTACION TEORICA.

El presente capítulo tiene el propósito de servir como base para la comprensión del resto de la investigación. Para ello, en él se describen las características fundamentales del módulo Consulta Externa del Sistema de Información Hospitalaria alas HIS y los procesos de negocio a informatizar. Además se definen las tecnologías, metodologías y herramientas informáticas a utilizar para este fin, seleccionadas por el departamento Gestión Hospitalaria para el desarrollo del Sistema de Información Hospitalaria alas HIS, debido a que el diseño a implementar formará parte de la Consulta Externa de Nefrología del módulo Consulta Externa de dicho sistema.

1.1 Descripción del módulo Consulta Externa del Sistema de Información Hospitalaria alas HIS.

La Consulta Externa es el departamento en el cual se brinda atención médica a los enfermos no internados y cuyo padecimiento les permite acudir al hospital. Dicha atención puede ser de diversa índole, pero principalmente consiste en el interrogatorio, la indicación de exámenes que conducen al diagnóstico, la prescripción de un tratamiento y de ser necesario, la remisión de un paciente para otras áreas de cualquier centro asistencial. [3]

Como parte del Sistema de Información Hospitalaria alas HIS, el módulo de Consulta Externa tiene como objetivo capturar la información de cada uno de los servicios que se prestan en las consultas externas de determinada unidad de salud. Puesto que los servicios están relacionados con especialidades médicas, este módulo cuenta con la Consultas Externas de Traumatología y Ortopedia, Neurología, Ginecología, Oftalmología, Dermatología y Estomatología, entre otras, pero no cuenta con el servicio de Nefrología.

1.2 Descripción del proceso relacionado con el trasplante renal en la Consulta Externa de Nefrología

Los pacientes que padecen de alguna enfermedad renal crónica se tratan en la Consulta Externa de Nefrología; donde el médico puede determinar si existe la necesidad de realizar un trasplante, en caso positivo le indicará al mismo que indague sobre la posibilidad de que algún familiar le done un riñón y le orientará que en caso afirmativo venga acompañado de esta(s) persona(s) a la próxima consulta.

Una vez llegado este momento, si el paciente alega no tener familiar alguno que pueda fungir como donante, el médico registrará al paciente en la lista de espera nacional de receptores renales si su estado de salud lo permite, es decir, si es apto para trasplante, si es no apto esperará a la próxima consulta para valorar su estado de aptitud y en caso posible registrarlo en dicha lista. La cirugía será programada cuando al paciente le corresponda su turno según la lista de espera nacional de receptores renales.

Si el paciente viene acompañado de algún posible donante, el médico nefrólogo indicará al donante y al receptor exámenes generales y específicos para determinar el grado de compatibilidad entre ellos. Una vez realizados los estudios a todos los implicados, el médico nefrólogo decidirá a partir de los resultados quién es el donante idóneo, es decir, el más compatible y con posibilidades de donar y se programará la cirugía si el paciente es apto para trasplante. Si no existiera un donante idóneo y el paciente es apto, entonces se registrará en la lista de espera nacional de receptores renales, si no es apto el médico nefrólogo esperará a la próxima consulta para valorar si puede registrarlo.

Cuando un paciente es trasplantado, tendrá que asistir a consultas de seguimiento de por vida. Si el trasplante no es exitoso, entonces se tratará al paciente como la primera vez que asistió a consulta en estado crónico de la enfermedad.

1.3 Conceptos asociados.

- *Enfermedad renal crónica:* pacientes que presentan pérdida de la capacidad que deben tener los riñones para eliminar desechos, concentrar la orina y conservar los electrolitos.[4]
- *Trasplante renal:* cirugía que se le realiza a un paciente para sustituir el riñón dañado por uno sano.[5]
- *Apto para trasplante:* paciente que presenta las condiciones necesarias para que le sea realizado el trasplante.
- *Donante idóneo:* persona ideal para fungir como donante renal según sus condiciones clínicas y de compatibilidad con el paciente.

1.4 Funcionalidades a implementar

Una vez analizados los procesos de negocio asociados con el trasplante renal en la Consulta Externa de Nefrología se identificaron las siguientes funcionalidades a implementar:

- Gestionar Hoja de trasplante renal
- Determinar donante idóneo
- Dar salida donante
- Buscar donante
- Registrar donante
- Gestionar Hoja nefrológica del donante

A partir de los cuales se definen los siguientes casos de uso del sistema:

- Buscar donante
- Registrar donante
- Seleccionar donante
- Crear hoja nefrológica del donante
- Ver hoja nefrológica del donante
- Modificar hoja nefrológica del donante
- Eliminar hoja nefrológica del donante
- Crear salida al donante del Registro de Trasplante.
- Crear hoja de trasplante renal
- Ver hoja de trasplante renal
- Modificar hoja de trasplante renal
- Eliminar hoja de trasplante renal
- Determinar donante idóneo

1.5 Metodología de desarrollo de software a emplear.

Para guiar el proceso de desarrollo del software se utilizó la metodología Proceso Unificado de Desarrollo.

1.5.1 Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo (por sus siglas en inglés RUP) es un proceso para el desarrollo de un proyecto de un software que define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto. Esta metodología tiene tres características fundamentales: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental.

Junto con el Lenguaje Unificado de Modelado (UML por sus siglas en inglés), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. [6]

Ventajas:

- Mitigación temprana de posibles riesgos altos.
- Progreso visible en las primeras etapas.
- Temprana retroalimentación que se ajuste a las necesidades reales.
- Gestión de la complejidad.
- El conocimiento adquirido puede aplicarse de iteración a iteración.

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.

1.6 Lenguaje Unificado de Modelado

El Lenguaje de Modelado Unificado (siglas en inglés UML) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar sistemas conceptuales como lo son procesos de negocio y funciones de sistema, además de elementos concretos como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. [7]

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño, donde se indican los pasos que se deben seguir para llegar a un diseño. La

estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático [8]

1.7 Tecnologías.

Java Server Faces

Java Server Faces (JSF) es un framework para aplicaciones Java de tipo web, que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE (Java Enterprise Edition). JSF usa Java Server Pages (JSP) como la tecnología que permite realizar el despliegue de las páginas.

JSF incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Dos librerías de etiquetas personalizadas para JSP que permiten expresar una interfaz JSF dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Administración de estados. [9]

Características

- Una clara separación entre la vista y la modelo.
- Desarrollo basado en componente, no en peticiones.
- Las acciones del usuario se ligan muy fácilmente al código en el servidor.
- Ofrece múltiples posibilidades de elección entre distintos desarrollos.

Ventajas

- Ofrece una clara separación entre el comportamiento y la presentación. Las aplicaciones web construidas con tecnología JSP conseguían parcialmente esta separación. Sin embargo, una aplicación JSP no puede mapear peticiones HTTP al manejo de eventos específicos de los componentes o manejar elementos interfaz de usuario como objetos con estado en el servidor.
- Permite construir aplicaciones web que implementan una separación entre el comportamiento y la presentación tradicionalmente ofrecida por arquitectura UI del lado del cliente. JSF se hace fácil de usar al aislar al desarrollador del API de Servlet.
- La separación de la lógica de la presentación proporciona un sencillo modelo de programación para enlazar todas las piezas.

- Proporciona una rica arquitectura para manejar el estado de los componentes, procesar los datos, validar la entrada del usuario, y manejar eventos.
- Ofrece una rápida adaptación para nuevos desarrolladores.[10]

Seam UI

Es un framework de código abierto que enlaza diferentes tecnologías y estándares Java. Esto garantiza la plena comunicación de los elementos de la capa de presentación, de negocio y de acceso a datos. Con Seam basta agregar anotaciones propias de éste a los objetos Entidad y Session de EJB, lo que permite escribir menos código Java y XML. Otra característica importante es que se puede hacer validaciones en los POJOs y además manejar directamente la lógica de la aplicación y de negocios desde las sessions bean. [11]

Ventajas

- Contextos: además de los contextos clásicos agrega contextos muy útiles sobre todo el contexto CONVERSATION.
- Manejo de excepciones y páginas de error: Facilita la administración de las excepciones y provee opciones de configuración para las páginas de error de la aplicación.
- Manejo de mensajes: Facilita el manejo de mensajes por medio de extensiones al FacesMessage de JSF e inyección del mismo.
- Validaciones: Con el uso de Hibernate-Validator
- Posee varios tipos de componentes: Interceptores, Beans, Logger, manejadores de eventos, FacesMessages, IdentityStore, etc.
- Bijection: Permite la clásica inyección y agrega la outyección
- Componentes visuales: Permite utilizar frameworks de componentes visuales como RichFaces. También provee tags especializados que facilitan muchas tareas.

RichFaces

RichFaces es una biblioteca de componentes para JSF y un avanzado framework para la integración de AJAX con facilidad en la capacidad de desarrollo de aplicaciones de negocio.

Ajax4jsf y RichFaces son bibliotecas de código abierto que se integran totalmente en la arquitectura de JSF y heredan las funcionalidades de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código JavaScript. Mediante este framework se puede variar el ciclo de vida de

una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones al servidor automáticas y controlar cualquier evento de usuario. En definitiva Ajax4jsf y RichFaces permiten dotar una aplicación JSF de contenido mucho más profesional con muy poco esfuerzo.

El funcionamiento del framework es sencillo. Mediante sus propias etiquetas se generan eventos que envían peticiones al contenedor Ajax. Estos eventos se pueden ejecutar por pulsar un botón, un enlace, una región específica de la pantalla, un cambio de estado de un componente. Esto significa que no hay que preocuparse de crear el código JavaScript y el objeto XMLHttpRequest para que envíe la petición al servidor, ya que el framework lo hará de forma automática. [12]

Características

- RichFaces está completamente integrado en el ciclo de vida de JSF.
- Añade capacidad Ajax a aplicaciones JSF.
- Los componentes de RichFaces están diseñados para ser usados sin problemas con otras librerías de componentes en la misma página, de modo que existen más opciones para el desarrollo de aplicaciones.
- Proporciona un paquete de recursos con clases de aplicación Java.

Ventaja

- Al pertenecer RichFaces a un subproyecto de JBoss, su integración con Seam es perfecta.

Ajax

Ajax es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que permite aumentar la interactividad, velocidad y usabilidad en las aplicaciones. Constituye una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM). [13]

Ventajas

- No se refresca la página constantemente al interactuar con ella.
- Reduce el tiempo de espera para una petición.
- El tráfico al servidor se reduce.

Ajax4jsf

Ajax4jsf es una librería de código abierto que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas, dotándolas con tecnología Ajax de forma limpia y sin añadir código JavaScript.

Ajax4jsf presenta mejoras sobre los propios beneficios del framework JSF donde incluye el ciclo de vida, validaciones, facilidades de conversión y el manejo de recursos estáticos y dinámicos. Permite definir un evento en una página, que invoca una petición Ajax y luego las áreas de la página deberían sincronizarse con el Árbol de Componentes JSF. [14]

Ajax4Jsf permite:

- Intensificar el conjunto de beneficios de JSF mientras se trabaja con AJAX.
- Añadir funcionalidad AJAX a las actuales aplicaciones JSF.
- Disponer de un paquete de recursos con las clases de la aplicación Java. Además de su núcleo, la funcionalidad de AJAX que se incluye en Ajax4jsf proporciona un avanzado apoyo a la gestión de diferentes recursos: imágenes, código JavaScript y hojas de estilo CSS.
- Soporte Ajax para páginas con componentes JSF que tienen comunicación asíncrona y actualizaciones de porciones de página.
- La capacidad de habilitar Ajax a componentes JSF sin cambios en el propio componente.
- Disponer de una arquitectura abierta y soporte para los estándares en la industria que permite mezclar componentes que son de distintas librerías.
- Trabajar en el lado del servidor.
- Soportar Facelets.

Ventaja

- La utilización de la funcionalidad AJAX dentro de páginas JSF sin la necesidad de crear nuevo código JavaScript.

Cascading Style Sheets (CCS)

Hojas de Estilo en Cascada (CCS por sus siglas en inglés), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. CSS se utiliza para dar estilo a documentos HTML y XML, donde se separa el

contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores web controlar el estilo y el formato de múltiples páginas al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a la CSS en las que aparezca ese elemento. [15]

Ventajas

- Mayor control de la presentación del sitio al poder tener todo el código CSS reunido en uno, lo que facilita su modificación.
- Aumenta la accesibilidad, ya que se le puede asignar un código CSS concreto a personas con deficiencias visuales.

Facelets

Framework de código abierto bajo la licencia de Apache. Soporta todos los componentes de interfaz de usuario JSF y construye su propio árbol de componentes, lo que refleja el punto de vista de una aplicación JSF, por lo que aporta mayor libertad a los desarrolladores pues permite definir vistas JSF mediante la utilización de plantillas del tipo HTML, reduce así el código innecesario para agregar componentes en la vista y que no necesariamente sea un contenedor web. Las principales ventajas de Facelets son:

- Construcción de interfaces basadas en plantillas.
- Rápida creación de componentes por composición.
- Fácil creación de funciones y librerías de componentes. [16]

Características

- En Facelets, las páginas son XTHML.
- Facelets permite incluir texto, etiquetas y expresiones en cualquier zona de la página, y se encargará de evaluarlo.
- Se mejora el manejo de mensajes de error, de esta manera se reduce, drásticamente el tiempo necesario para interpretar un error en JSF.
- Permite crear componentes ligeros. [17]

XHTML

Lenguaje de Marcado de Hipertexto Extensible, es una versión más estricta y limpia de HTML. XHTML surge como el lenguaje cuyo etiquetado, más estricto que HTML, va a permitir una correcta interpretación de la información independientemente del dispositivo desde el que se accede a ella.

XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos. [18]
XHTML puede incluir otros lenguajes como MathML, SMIL o SVG, al contrario que HTML.
XHTML, al estar orientado al uso de un etiquetado correcto, exige una serie de requisitos básicos a cumplir en lo que a código se refiere. Entre estos requisitos básicos se puede mencionar una estructuración coherente dentro del documento donde se incluirían elementos correctamente anidados, etiquetas en minúsculas, elementos cerrados correctamente y atributos de valores entrecomillados. [19]

JBoss Seam

JBoss Seam es un framework que integra y unifica los distintos estándares de la plataforma Java EE 5.0, pudiendo trabajar con todos ellos siguiendo el mismo modelo de programación. El núcleo principal de Seam está formado por las especificaciones Enterprise JavaBeans 3 (EJB3) y JSF. [20]
Algunos de los framework con los que se integra son:

- Java Server Faces (JSF): Para el desarrollo de la interfaz de usuario, en este sentido Seam contiene la librería RichFaces/Ajax4jsf, que reduce enormemente el esfuerzo de los programadores con los componentes web.
- Hibernate: Para el mapeo y el acceso con la base de datos. [21]

Hibernate

Hibernate es una herramienta de mapeo objeto relacional (ORM) para la plataforma Java (disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.
Hibernate ofrece también un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), el cual solo difiere con SQL (Structured Query Language) en cuanto a que es completamente orientado a objetos y comprende nociones como herencia, polimorfismo y asociación. [22]
Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación. Para lograr esto permite al

desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Hibernate está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible. [23]

Enterprise JavaBeans (EJB)

Los Enterprise JavaBeans (EJB) son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE de Oracle Corporation. Una de las metas de la arquitectura EJB es la de poder escribir de manera fácil aplicaciones de negocio orientadas a objetos y distribuidas, basadas en el lenguaje de programación Java. El propósito de EJB es el de proveer el soporte de la arquitectura de EJB y al mismo tiempo, reducir la complejidad para el desarrollo de aplicaciones empresariales.[24]

Ventajas

- Portabilidad de la aplicación: Una aplicación EJB puede ser desplegada en cualquier servidor de aplicaciones que soporte J2EE.
- Reusabilidad de componentes: Una aplicación EJB está formada por componentes Enterprise Beans. Cada Enterprise Bean es un bloque de construcción reusable.
- Posibilidad de construcción de aplicaciones complejas: La arquitectura EJB simplifica la construcción de aplicaciones complejas.
- Separación de la lógica de presentación de la lógica de negocio: Un Enterprise Bean encapsula típicamente un proceso o una entidad de negocio.
- Despliegue distribuido: La arquitectura EJB hace posible que las aplicaciones se desplieguen de forma distribuida entre distintos servidores de una red.[25]

Java Persistence API

Java Persistence API (JPA) proporciona un modelo de persistencia basado en POJOs para mapear bases de datos relacionales en Java. El Java Persistence API fue desarrollado por el grupo de expertos de EJB 3.0 como parte de JSR 220, aunque su uso no se limita a los componentes software EJB. También puede utilizarse directamente en aplicaciones web y aplicaciones clientes; incluso fuera de la plataforma Java EE, por ejemplo, en aplicaciones Java SE. [26]

JBoss Application Server

JBoss Application Server es un servidor de aplicaciones J2EE implementado en Java puro. JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible Java. Implementa todo el paquete de servicios de J2EE y es el primer servidor de aplicaciones de código abierto preparado para la producción y certificado. [27]

Java Runtime Environment (JRE)

Java Runtime Environment (JRE) es un conjunto de herramientas necesarias para el funcionamiento de cualquier aplicación que haya sido desarrollada a partir del lenguaje Java. Por eso, ya se trate de un profesional o un usuario doméstico, este es un software que no debe faltar en el ordenador.

JRE es compatible con la mayoría de los navegadores web, como Internet Explorer, Mozilla Firefox, Netscape Navigator, entre otros. Por otra parte, posee también el beneficio de ser multiplataforma y estar disponible en diversos idiomas. [28]

PostgreSQL 8.3

PostgreSQL es un sistema gestor de base de datos que se destaca por su robustez, escalabilidad y cumplimiento de los estándares SQL. Este cuenta con diversas versiones para sistemas operativos tales como: Linux, Windows, Unix, Mac OS X, Solaris, BSD, Tru64, entre otros.

Soporta vistas, uniones, claves extranjeras, triggers, incluye la mayor parte de los tipos de datos especificados en los estándares SQL92 y SQL99 y presenta soporte de protocolo de comunicación encriptado por SSL, extensiones para alta disponibilidad, nuevos tipos de índices y minería de datos. Implementa internamente lenguajes de consulta de muy alto nivel como son el plpgsql (muy similar al plsql de Oracle), el plperl, el plpyton y el c, ejecuta consultas y scripts SQL, visualiza y edita datos, representa datos como diagramas, exporta e importa datos desde y hacia los formatos de archivos de uso más popular. [29]

Además administra roles, usuarios, grupos y sus privilegios, y usa una serie de herramientas adicionales diseñadas para una fácil y eficiente operación con el servidor PostgreSQL.

Dentro de las características que se destacan de PostgreSQL y que lo convierten en una herramienta eficiente para el trabajo en las base de datos están la atomicidad, la consistencia y la durabilidad, juntas aseguran que solo empieza aquello que se puede acabar, garantizan que una operación no puede afectar a otras y una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema. [30]

1.8 Plataforma de desarrollo.

Java Platform Enterprise Edition (JavaEE 5)

Java Platform Enterprise Edition o Java EE es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java con arquitectura de n niveles distribuidos, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE

Algunos beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, lo que significa que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel. [31]

1.9 Herramientas informáticas.

Eclipse

Eclipse es un entorno de desarrollo integrado (IDE por sus siglas en inglés) portable y multiplataforma. Brinda una plataforma universal para integrar herramientas de desarrollo.

La arquitectura de plug-ins permite integrar diversos lenguajes sobre un mismo IDE e introducir otras aplicaciones. Hay plug-ins para el desarrollo de Java (JDT Java Development Tools), así como para el desarrollo en C/C++, COBOL, estos conservan el registro de las versiones, generan y mantienen la documentación de cada etapa del proyecto.

Características.

- Posee un editor visual con sintaxis coloreada.
- Permite la compilación incremental de código.
- Modifica e inspecciona valores de variables. [32]

PgAdmin

PgAdmin es una aplicación gráfica que se utiliza para interactuar con el gestor de bases de datos PostgreSQL. Está escrita en C++, usa la librería gráfica multiplataforma wxWidgets, lo que aprueba que se pueda usar en Linux. Presenta una interfaz gráfica amigable con todas las características de PostgreSQL, lo que facilita grandemente la administración de la base de datos. La aplicación incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, entre otras

funcionalidades que hacen de esta una de las herramientas más usadas con licencia Open Source en la actualidad. [33]

Ventajas

- Costo de adquisición bajo o nulo.
- Cero problemas de licencias.
- Si se usa en Linux, es inmune a los virus.
- Requerimientos mínimos para instalación. [34]

Visual Paradigm

Visual Paradigm es una eficaz herramienta para visualizar y diseñar elementos de software, para ello utiliza UML y ofrece una gama de facilidades para el modelado de aplicaciones.

Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. Ofrece una ayuda para la construcción más rápida de aplicaciones de calidad, ya que permite dibujar todos los tipos de diagramas de clases y provee soporte para la generación de código.

Tiene integración con diversos IDE's como NetBeans (de Sun Microsystems), JDeveloper (de Oracle), Eclipse (de IBM), JBuilder (de Borland), así como la posibilidad de realizarse la ingeniería inversa para aplicaciones realizadas en Java, .NET, XML e Hibernate.

También tiene disponibilidad en múltiples plataformas y soporta BPMN (Business Process Modeling Notation). [35]

Conclusiones

En este capítulo se describió el módulo Consulta Externa del Sistema de Información Hospitalaria alas HIS, se puntualizó el proceso de trasplante renal en la Consulta Externa de Nefrología y se expusieron conceptos relacionados con el modelo de dominio, lo que contribuye a un mejor entendimiento de esta investigación. Se analizaron el conjunto de herramientas y tecnologías, así como la metodología a utilizar en la implementación de la Hoja de trasplante renal del componente Consulta Externa de Nefrología del Sistema de Información Hospitalaria alas HIS.

Se utilizaron las siguientes tecnologías y herramientas:

- Java Server Faces
- Seam UI
- RichFaces

- Ajax
- Ajax4jsf
- Cascading Style Sheets (CCS)
- Facelets
- XHTML
- JBoss Seam
- Hibernate
- Java Persistence API
- JBoss Application Server
- Lenguaje Unificado de Modelado
- PostgreSQL 8.3
- Eclipse
- PgAdmin
- Java Platform Enterprise Edition
- Visual Paradigm

Esta propuesta tecnológica fue definida por el Departamento de Gestión Hospitalaria para el desarrollo de sus aplicaciones y será utilizada en la presente investigación dado que el resultado de la misma formará parte de uno de sus módulos.

CAPÍTULO 2: DESCRIPCION DE LA ARQUITECTURA

Este capítulo tiene como propósito fundamental describir la arquitectura del software a desarrollar, para que se logre una mejor comprensión de los aspectos técnicos de la presente investigación. Para ello se describen los requerimientos no funcionales, los aspectos relacionados con la seguridad, la vista de despliegue, las estrategias de codificación, los estándares y estilos a utilizar.

2.1 Requerimientos no funcionales.

2.1.1 Usabilidad

El sistema estará diseñado de manera que los usuarios adquieran las habilidades necesarias para explotarlo en un tiempo reducido:

Usuarios normales: 30 días

Usuarios avanzados: 20 días

2.1.2 Fiabilidad

Se mantendrá seguridad y control a nivel de usuario, lo que garantiza el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realizan. Las contraseñas podrán cambiarse solo por el propio usuario o por el administrador del sistema.

Con el objetivo de elaborar reportes estadísticos por parte del hospital o entidad, ninguna información que se haya ingresado en el sistema será eliminada físicamente de la BD, independientemente de que para el sistema, este elemento ya no exista. También es configurable el tiempo que los datos perduran en la BD. A estos datos se pueden acceder mediante Visor portable de la Historia Clínica que es capaz de extraer todos los datos de un paciente determinado y grabarlos en un CD.

2.1.3 Eficiencia

El sistema respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos. Se deberá usar siempre que sea posible el patrón Singleton, este se implementa mediante el framework SEAM, el cual proporciona un conjunto de anotaciones, de las cuales se usa @Scope, el cual crea una instancia diferente de la clase por cada sesión Http. Además se debe destruir referencias que ya no estén siendo usadas, optimizar el trabajo con cadenas, entre otras buenas prácticas que ayudan a mejorar el rendimiento.

2.1.4 Soporte

Abarcan las acciones que se llevarán a cabo una vez que se ha terminado el desarrollo del software, con motivos de asistir a los clientes de este, lo que trae como consecuencia una mejora progresiva y evolución en el tiempo. Pueden incluir: pruebas, extensibilidad, adaptabilidad, mantenimiento, configuración, compatibilidad, servicios e instalación.

Las notificaciones de las deficiencias detectadas en la aplicación desplegada deberán realizarse por escrito.

Una vez notificada por la entidad médica, la deficiencia detectada en la aplicación desplegada, el equipo de desarrollo deberá solucionarla en un período de 7 días.

La capacitación y entrenamiento del profesional de salud para el uso del sistema, se realizará en un período de 6 meses.

Configuración de parámetros.

Se permitirá establecer parámetros de configuración del sistema y actualización de nomencladores.

2.2 Restricciones de diseño

El sistema estará dividido en las siguientes capas:

- **Capas Físicas**

Cliente: Computadora con sistema operativo Windows XP o GNU/Linux que cuente con un navegador actualizado y que siga los estándares web (se recomienda IE 7 o superior o Firefox 3.x).

Servidor de Aplicaciones: Servidor con sistema operativo GNU/Linux que soporte el Java Runtime Environment (JRE) 1.5 o superior y al JBoss AS 4.2 o superior.

Servidor de Base de Datos: Servidor con sistema operativo Windows XP o GNU/Linux que soporte a PostgreSQL Server 8.3 o superior en los servidores de base de datos de cada hospital

- **Capas Lógicas**

Presentación: Contiene todas las vistas y la lógica de la presentación. El flujo web se maneja de forma declarativa y basándose en definiciones de procesos del negocio.

Negocio: Mantiene el estado de las conversaciones y procesos del negocio que concurrentemente pueden estar siendo ejecutados por cada usuario. En los casos de que algún objeto del negocio tenga una interfaz externa, siendo accesible la misma desde sistemas legados o directamente del cliente, se garantiza la seguridad a nivel de objeto y métodos.

Acceso a Datos: Contiene las entidades y los objetos de acceso a datos correspondientes a las mismas. El acceso a datos está basado en el estándar JPA y particularmente en la implementación del motor de persistencia Hibernate.

2.2.1 Interfaz

Interfaces de usuario

- Las ventanas del sistema contendrán claro y bien estructurados los datos, además de permitir la interpretación correcta de la información.
- La interfaz contará con teclas de función y menús desplegables que faciliten y aceleren su utilización.
- La entrada de datos incorrecta será detectada claramente e informada al usuario.
- El sistema permitirá al usuario configurar el idioma de los mensajes que serán mostrados.
- El diseño de la interfaz del sistema responderá a la ejecución de acciones de una manera rápida, lo que minimiza los pasos a dar en cada proceso. El sistema incluirá reportes estándares y parametrizables que permitirán al usuario configurar la información de salida y el orden en que aparecen los datos. Las salidas se podrán generar en el formato de fichero PDF.

2.3 Descripción de la arquitectura.

A continuación se describe el estilo arquitectónico a utilizar para el desarrollo de la Hoja de trasplante renal del componente de Consulta Externa de Nefrología, perteneciente al módulo Consulta Externa del Sistema de Información Hospitalaria alas HIS. Dicho estilo fue seleccionado por el departamento Gestión Hospitalaria para la realización de dicho sistema.

Modelo Vista Controlador (MVC): es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos: el modelo, que abarca los datos y las reglas del negocio; la vista, que muestra la información del modelo al usuario; y el controlador, que gestiona las entradas del mismo. [36]

- *Modelo:* accede a la capa de almacenamiento de datos o acceso a datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento. Lleva un registro de las vistas y controladores del sistema. Ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.
- *Vista:* presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

- *Controlador:* responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

Ventajas que brinda el Modelo Vista Controlador:

- Es posible tener diferentes vistas para un mismo modelo.
- Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.
- Proporciona un mecanismo de configuración a componentes complejos mucho más tratables que el puramente basado en eventos. [37]

La capa de presentación está formada por páginas XHTML, las cuales están compuestas por formularios y controles JSF y RichFaces, que se encargan de validar y mostrar los datos donde hace uso del componente Ajax4jsf.

La capa de negocio está compuesta por clases controladoras, que definen la lógica del negocio conjuntamente con los datos que se validan en la capa de presentación. Todo esto se desarrolla mediante el framework Seam.

La capa de acceso a datos se maneja mediante Hibernate, que permite seleccionar, modificar, eliminar y persistir la información en la base de datos. Esto permite llevar las consultas a un lenguaje orientado a objetos. [38]

2.4 Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser reusados. Estrategias de integración.

El Sistema de Información Hospitalaria alas HIS, el cual se encuentra en desarrollo, está compuesto por diferentes módulos que comparten componentes entre sí. Para el desarrollo de la Hoja de trasplante renal, se utilizaron los siguientes módulos asociados con la implementación de los procesos enmarcados en el campo de acción del presente trabajo de diploma:

- *Módulo Laboratorio:* Vital importancia tienen en la especialidad de Nefrología los resultados de los múltiples estudios complementarios que se le realizan al paciente. Todos estos estudios se realizan en el módulo Laboratorio y el médico nefrólogo puede visualizar los resultados.
- *Módulo Consulta Externa:* Es de donde parte la Consulta Externa de Nefrología.

Semanalmente el arquitecto del proyecto agrupa las soluciones, que luego cada quince días son despachadas en el departamento Gestión Hospitalaria.

2.5 Seguridad del sistema

Con el objetivo de garantizar la confidencialidad, integridad y disponibilidad de la información de los pacientes en la aplicación, es necesario establecer un fuerte nivel de seguridad. Por tal motivo se ha definido un control de acceso a nivel de usuarios y contraseñas, con lo cual el usuario solo tendrá acceso a los niveles que le competan.

La autorización a directorios, páginas, controles, opciones del menú, servicios del negocio, está basada en reglas del negocio, dichas reglas no se encuentran en el código de la aplicación, lográndose de esta forma una total independencia con respecto a este y lo que posibilita que cualquier cambio de las reglas del negocio, no afecte en lo absoluto al código fuente. Lo anteriormente expuesto es posible gracias a la integración existente entre Seam Security Framework y el potente motor de reglas JBoss Rules.

2.6 Vista de Despliegue.

La vista de despliegue (Ver Figura 2.1) permite representar la distribución física del sistema, así como los distintos nodos (recursos de ejecución: servidores, computadoras o dispositivos tales como impresoras y scanners) relacionados por enlaces de comunicación. Modela los aspectos del hardware de un sistema, para que el ingeniero de software pueda especificar la plataforma sobre la que se ejecutará. A continuación se muestra el Diagrama de Despliegue del sistema.

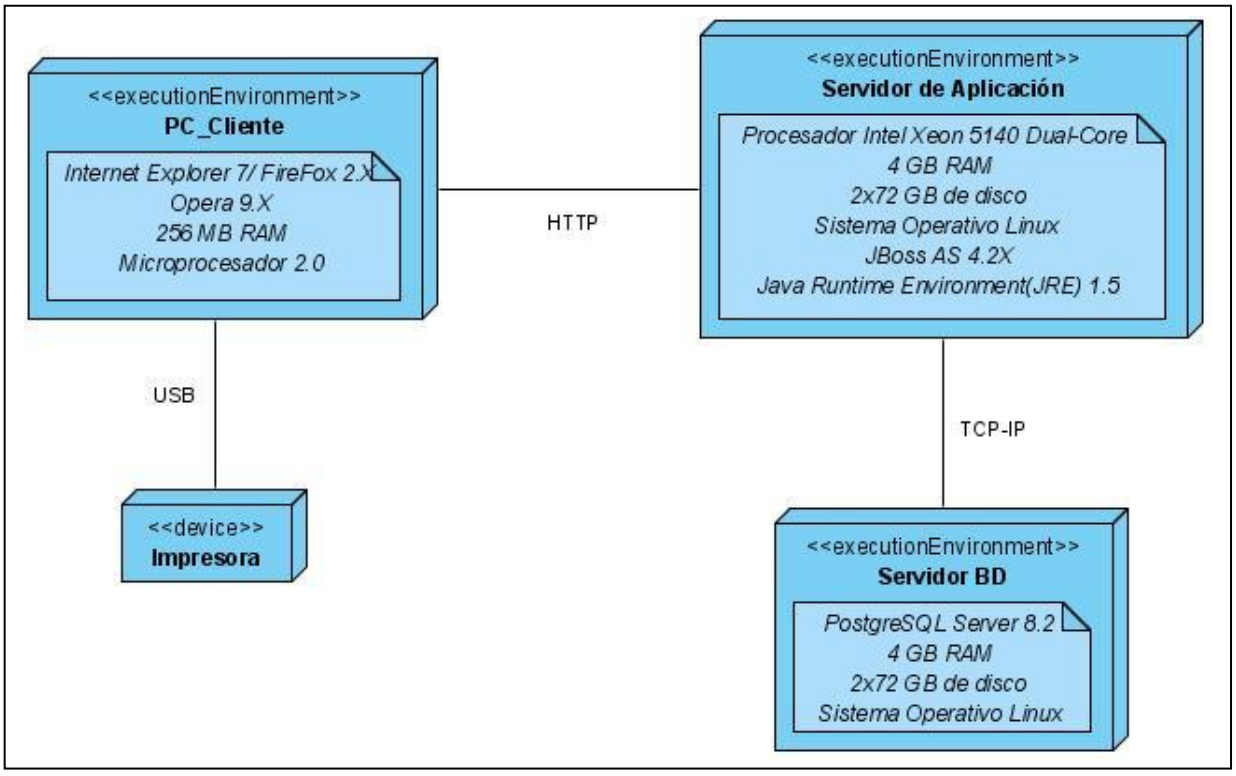


Figura 2.1 Diagrama de Despliegue

2.7 Estrategias de codificación. Estándares y estilos a utilizar.

Los estándares de codificación establecen un conjunto de reglas que los desarrolladores deben seguir para escribir el código fuente de un software. Esto tiene como objetivo que dicho código sea entendible por cualquier desarrollador del grupo de trabajo, garantizando un mantenimiento del sistema más rápido y eficiente, ya sea creando nuevas funcionalidades o modificando las ya existentes.

Para el desarrollo del sistema propuesto se utilizan varios estándares de codificación, tales como:

Notación Camell: se utiliza para denotar variables, parámetros y métodos. Si el identificador es una palabra simple se escribe todo con minúscula, en caso de que sea compuesta, se escribe con minúscula la primera letra del identificador y las que vienen a continuación con mayúscula.

A continuación se muestran algunas reglas de codificación a utilizar:

- **Indentación**

  Inicio y bloque de fin: Se debe dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque ({ }). Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach.

  Aspectos generales: El identado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la PC o la configuración de dicha tecla. Los inicios ({) y cierres (}) de ámbito deben estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción.

- **Comentarios, separadores, líneas, espacios en blanco y márgenes.**

  Ubicación de comentarios: Se debe comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de datos y objetivo del parámetro).

  Líneas en blanco: Se debe dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función.

  Espacios en blanco: Se deben usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código.

  Aspectos generales: Se debe evitar comentar cada línea de código; cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de

que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción. Con respecto a los espacios en blanco, no se deben usar: después del corchete abierto y antes del cerrado de un arreglo, después del paréntesis abierto y antes del cerrado ni antes de un punto y coma.

- **Variables y Constantes.**

Apariencia de constantes: Se deben declarar las constantes con todas sus letras en mayúscula.

Aspectos generales: El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.

- **Clases y Objetos**

Apariencia de clases y objetos: Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación Pascal Ejemplo: MiClase(). Para el caso de las instancias se comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.

Apariencia de atributos: El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará notación Camello.

Declaración de parámetro en funciones: Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que contienen, especificando el tipo de datos. (Ver Tabla 2.7.1)

Aspectos generales: El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.

- **Bases de datos, tablas, esquemas y campos**

Apariencia de la base de datos: Los nombres de las Bases de Datos deben comenzar con el prefijo bd a continuación underscoard y luego el nombre completamente en minúscula. Los nombres serán cortos y descriptivos.

Apariencia de las vistas: El nombre a emplear para las vistas deben comenzar con el prefijo vt seguido de underscoard y el nombre debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor.

Apariencia de las tablas: El nombre a emplear para las tablas debe comenzar con el prefijo tb seguido de underscoard y luego debe escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizará underscoard para separarlo.

Tablas que representen Relaciones: El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tr seguido de underscoard y el nombre de la tabla será la concatenación del nombre de las dos tablas que la generaron separados por uderscoard todo en minúscula.

Tablas que representen nomencladores: El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tn seguido de underscoard. El nombre será corto y descriptivo, todo en minúscula.

Apariencia de los procedimientos almacenados: El nombre a emplear para los procedimientos debe comenzar con el prefijo pa seguido de underscoard y luego debe escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizará underscoard para separarlo.

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.

Nombre de los campos: Todos los campos identificadores van a comenzar con el identificador id seguido de underscoard y posteriormente el nombre del campo.

Sentencias SQL: Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas.

Aspectos generales: El 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.

- **Controles**

Apariencia de los controles: El nombre que se le da a los controles deben comenzar con las primeras letras en minúscula, las cuales identificarán el tipo de dato al que se refiere (Ver Tabla 2.7.2), en caso de que sea un nombre compuesto se empleará notación CamellCasing**.

Tipo Datos	**Prefijo**	**Ejemplo**
Int	i	iCantPacientes
float	f	fPesoPaciente
double	d	dPesoCarro
bool	b	bPacienteActivo
string	s	sNombrePaciente
char	c	cLetra
De tipo enum	e	eSexo
byte	b	bCantDiasPaciente
sbyte	sb	sbEdadPaciente
short	sh	shVariableShort
ushort	us	usVariableUshort
uint	ui	uiVariableUint
long	l	lVariableLong
ulong	ul	ulVariableUlong
decimal	dc	dcVariableDecimal
Objetos	o	oPacienteHistorico
Objetos de tipo Struct	st	stUnaStruct

Tabla: 2.7.1 Tipo de datos

Control	**Prefijo**	**Ejemplo**
Botón	btn	btnAceptar
Etiqueta	lbl	lblNombre
Lista/Menú	mn	mnPrincipal
Campo de Texto	txt	txtFecha
Botón de Opción	bpt	optSexo
Casilla de Verificación	chx	chxBorrar

Tabla 2.7.2 Control

Conclusiones

En este capítulo se documentó la arquitectura del sistema, la cual constituye la base para llevar a cabo una implementación organizada y cuyo resultado sea un software robusto. Además, servirá como guía para futuras mejoras o adiciones que se le quieran realizar a la Hoja de trasplante renal del componente Consulta Externa de Nefrología del Sistema de Información Hospitalaria alas HIS.

CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA.

A partir de la arquitectura descrita en el capítulo anterior, resulta más fácil la implementación de un sistema que cumpla con los requisitos funcionales y no funcionales especificados. El presente capítulo tiene como objetivo refinar el diseño propuesto por el analista para cubrir las necesidades de la fase de implementación.

3.1 Valoración crítica del diseño propuesto por el analista.

Con el objetivo de propiciar una mejor comprensión y calidad del diseño, fueron aplicados patrones para dar solución a problemas comunes que se presentan en el diseño de aplicaciones; facilitándose la reusabilidad, extensibilidad y mantenimiento; economizando tiempo y optimizando el software, haciéndolo más eficiente, dinámico y seguro.

Dentro de los patrones utilizados se encuentran los GRASP, patrones para asignar responsabilidades, que tuvieron una importante utilidad. A cada clase le fueron asignadas las tareas que podían realizar según la información que poseía, además de crear las instancias de otras clases en correspondencia con la responsabilidad dada, poniéndose de manifiesto los patrones Experto y Creador respectivamente, siendo el primero uno de los más usados. Con esto se logró conservar el encapsulamiento, pues los objetos logran valerse de su propia información para cumplir con sus responsabilidades.

El diseño obtenido cumple con los patrones de Bajo Acoplamiento y Alta Cohesión, que permite la colaboración entre los elementos del diseño (clases), sin verse afectados la reutilización de los mismos y el entendimiento de estos cuando se encuentran aislados. La creación de clases controladoras facilitó realizar las operaciones del sistema, debido a que estas reflejan los procesos del negocio, los cuáles no resulta factible que se manejen en la capa de interfaz o presentación.

La utilización del patrón Modelo-Vista-Controlador permitió la separación de las capas de interfaz, negocio y datos. Permite una organización entre las clases de acuerdo a su funcionamiento, donde las clases encargadas del acceso a datos son las modelos, las clases encargadas de la visualización de la información son las vistas y las clases encargadas del flujo de datos entre las vistas y las modelos son las controladoras.

Una vez analizado el diseño propuesto por el analista, se determinó la necesidad de añadir nuevas tablas a la base de datos, dentro de las que se encuentran: tb_ nom_estado_aptitud, tb_nom_resultado, tb_tipo_trasplante y tb_parentesco. Estas nuevas tablas implicaron la generación de nuevas clases entidades. Para gestionar estas entidades se crearon sus respectivas clases modelos, controladoras y vistas.

3.2 Descripción de las nuevas clases u operaciones necesarias.

Nombre: Tbn_parentesco	
Tipo de clase: Entidad	
Atributo	**Tipo**
id	Integer
versión	Integer
código	String
valor	String
cid	Integer
eliminado	boolean
Para cada responsabilidad:	
Nombre:	TbnParentesco
Descripción:	Nomenclador

Tabla 3.2.1 Tbn_parentesco

Nombre: Buscar_donante	
Tipo de clase: Controladora	
Atributo	**Tipo**
donanteSeleccionado	String
donanteSeleccionado	String
donanteSeleccionadoOtro	String
hojaFrontal	HojaFrontal_consulta
busquedaAvanzada	String
busqueda simpre	String
busquedaAvanzadaStyle	String
inicioSesion_consulta	InicioSesion_consulta
orden	String
idPaciente	Integer
listado	List<HojaFrontal_consulta>
Para cada responsabilidad:	
Nombre:	Buscar_donante
Descripción:	Lista los donantes del paciente en consulta

Tabla 3.2.2 Buscar_donante

Nombre: AdicionarPaciente_consulta	
Tipo de clase: Controlodora	
Atributo	**Tipo**
tparentesco	String
pacienteSelec	HojaFrontal_consulta
donante	TbPacienteDonante_consulta
hojaFrontal	HojaFrontal_consulta
parentescoDonante	TbnParentesco_consulta
pacienteDonante	TbPacienteDonanteId_consulta
inicioSesion_consulta	InicioSesion_consulta
Para cada responsabilidad:	

Nombre: AdicionarPaciente_consulta	
Nombre:	Buscar_donante
Descripción:	Lista los donantes del paciente en consulta

Tabla 3.2.3 AdicionarPaciente_consulta

Nombre: Buscar_paciente	
Tipo de clase: Controladora	
Atributo	**Tipo**
donanteSeleccionado	String
donanteSeleccionado	String
donanteSeleccionadoOtro	String
hojaFrontal	HojaFrontal_consulta
busquedaAvanzada	String
busqueda simpre	String
busquedaAvanzadaStyle	String
inicioSesion_consulta	InicioSesion_consulta
orden	String
idPaciente	Integer
listado	List<HojaFrontal_consulta>
Para cada responsabilidad:	
Nombre:	Buscar_donante
Descripción:	Lista los donantes del paciente en consulta

Tabla 3.2.4 Buscar_paciente

3.3 Modelo de datos.

Un modelo de datos (Ver Figura 3.1) es un conjunto de conceptos, reglas y convenciones que permiten describir y en ocasiones manipular los datos de un cierto mundo real que se desea almacenar en la base de datos. Al producto del modelo de datos se le llama esquema (descripción de la estructura de la base de datos) y a los datos en concreto almacenados en la base de datos en ese momento, ocurrencia del esquema.

El modelo de datos está formado por dos componentes: componente estática, relacionada con el Lenguaje de Definición de Datos (LDD) y dinámica, relacionada con el Lenguaje de Manipulación de Datos (LMD). La parte estática se refiere a la estructura y la dinámica a qué operaciones se puede realizar sobre cada objeto. La parte estática del modelo de datos define:

- Objetos (entidades, relaciones)
- Asociaciones entre objetos (interrelaciones)
- Propiedades de los objetos (atributos, campos)
- Dominios: conjunto de valores del que se toman los valores para las propiedades, por ejemplo el dominio de los días de la semana para una propiedad día de nacimiento.

- Restricciones: Limitaciones impuestas por el modelo de datos que se utilice (inherentes) o definidas por el diseñador para representar fielmente el mundo real (de integridad o semánticas). [39]

A continuación se muestra el Modelo de datos perteneciente al componente Consulta Externa de Nefrología del módulo Consulta Externa, del Sistema de Información Hospitalaria alas HIS.

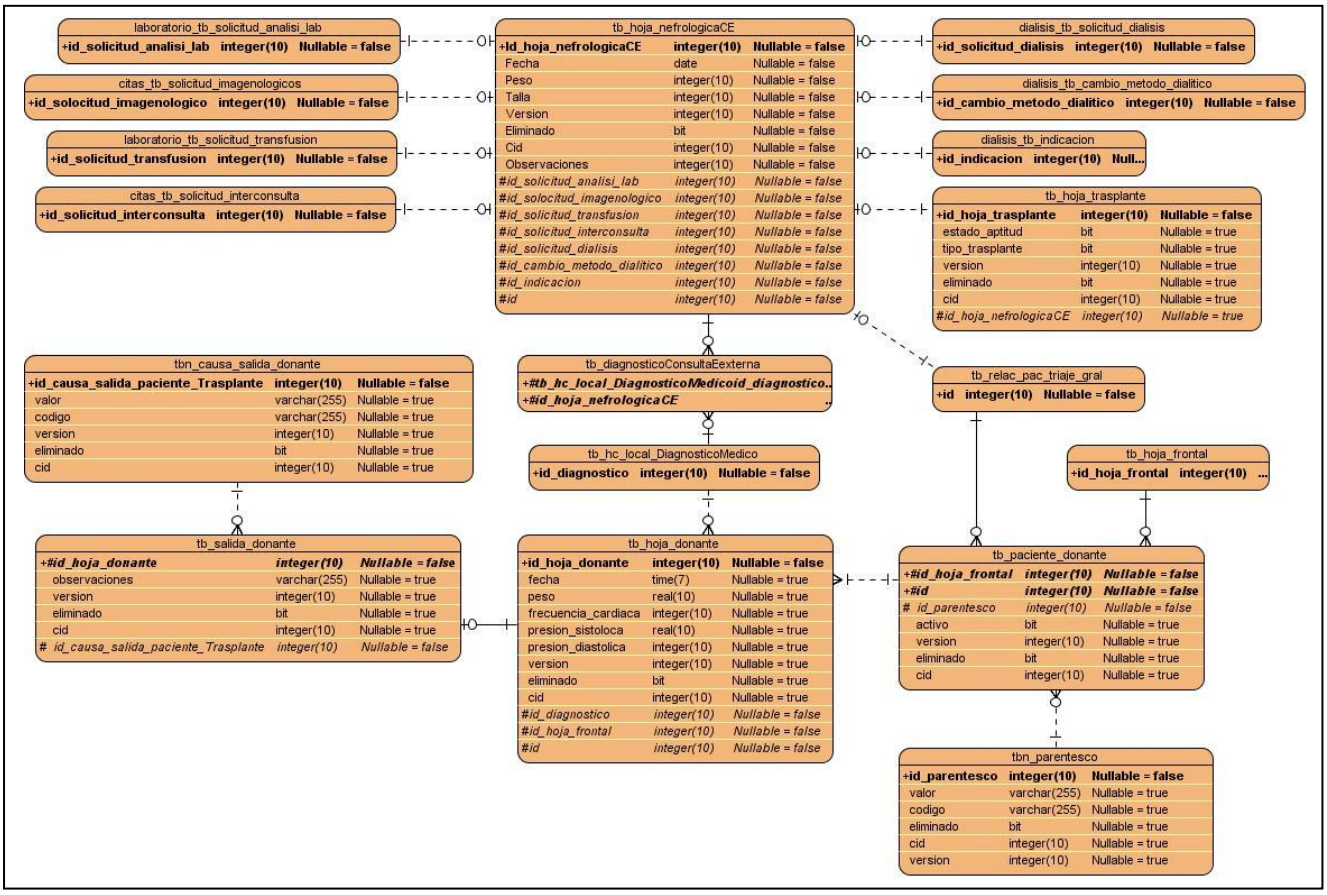


Figura 3.1 Modelo de datos del componente Consulta Externa de Nefrología

3.4 Descripción de las tablas.

TbHojaTrasplante	
Atributos	**Descripción**
Id_hoja_trasplante	Identificador de la hoja de trasplante
Id_hoja_nefrologicaCE	Identificador de la hoja nefrológica
tbHojaDonante	Objeto de la tabla de hoja donante
nomTipoTrasplante	Nomenclador tipo de trasplante que se llevará a cabo
nomEstadoAptitud	Nomenclador estado de aptitud del receptor
versión	Versión
eliminado	Campo eliminado

TbHojaTrasplante	
cid	Identificador
observaciones	Observaciones que se le realizaron al paciente

Tabla: 3.4.1 TbHojaTrasplante

TbHojaDonante	
Campo	**Descripción**
id_hoja_donante	Identificador del donante
id	Identificador de la relación paciente triaje general
id_hoja_frontal	Identificador de la hoja frontal de consulta externa
id_diagnostico	Identificador del diagnostico
fecha	Fecha de entrada del donante
peso	Peso del donante
frecuencia_cardiaca	Frecuencia cardiaca del donante
presion_sistolica	Presión sistólica
presion_diastolica	Presión diastólica
versión	Versión
eliminado	Campo eliminado
cid	Identificador

Tabla: 3.4.2 TbHojaDonante

TbPacienteDonante	
Atributos	**Descripción**
id_hoja_frontal	Identificador de la hoja frontal de consulta externa
Tbn_parentesco	Identificador del parentesco
fecha_salida	Fecha de salida del donante
versión	Versión
eliminado	Campo eliminado
cid	Identificador

Tabla: 3.4.3 TbPacienteDonante

TbCausaSalidaDonante	
Atributos	**Descripción**
id	Identificador de la causa salida donante
versión	Versión
código	Código
valor	Valor de la causa salida
cid	Cid
eliminado	Campo eliminado

Tabla: 3.4.4 TbCausaSalidaDonante

NomTipoTrasplante_consulta

NomTipoTrasplante_consulta	
Atributos	**Descripción**
id	Identificador de la causa salida donante
versión	Versión
código	Código
valor	Valor de la causa salida
cid	Cid
eliminado	Campo eliminado

Tabla: 3.4.5 NomTipoTrasplante_consulta

3.5 Vista de implementación. Diagrama de Componentes.

La vista de implementación muestra el empaquetado físico de las partes reutilizables del sistema en unidades sustituibles, llamadas componentes, que no son más que piezas reutilizables de alto nivel a partir de las cuales se pueden construir diferentes sistemas. Esta vista tiene como propósito definir la organización del código, planificar las integraciones del sistema necesarias en cada iteración e implementar las clases y subsistemas definidos durante el diseño. Para ilustrar la vista de implementación estática se construyen diagramas de componentes haciendo uso del lenguaje UML.

El diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables o paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

A continuación se muestra el Diagrama de Componentes del componente Consulta Externa de Nefrología del módulo Consulta Externa, del Sistema de Información Hospitalaria alas HIS

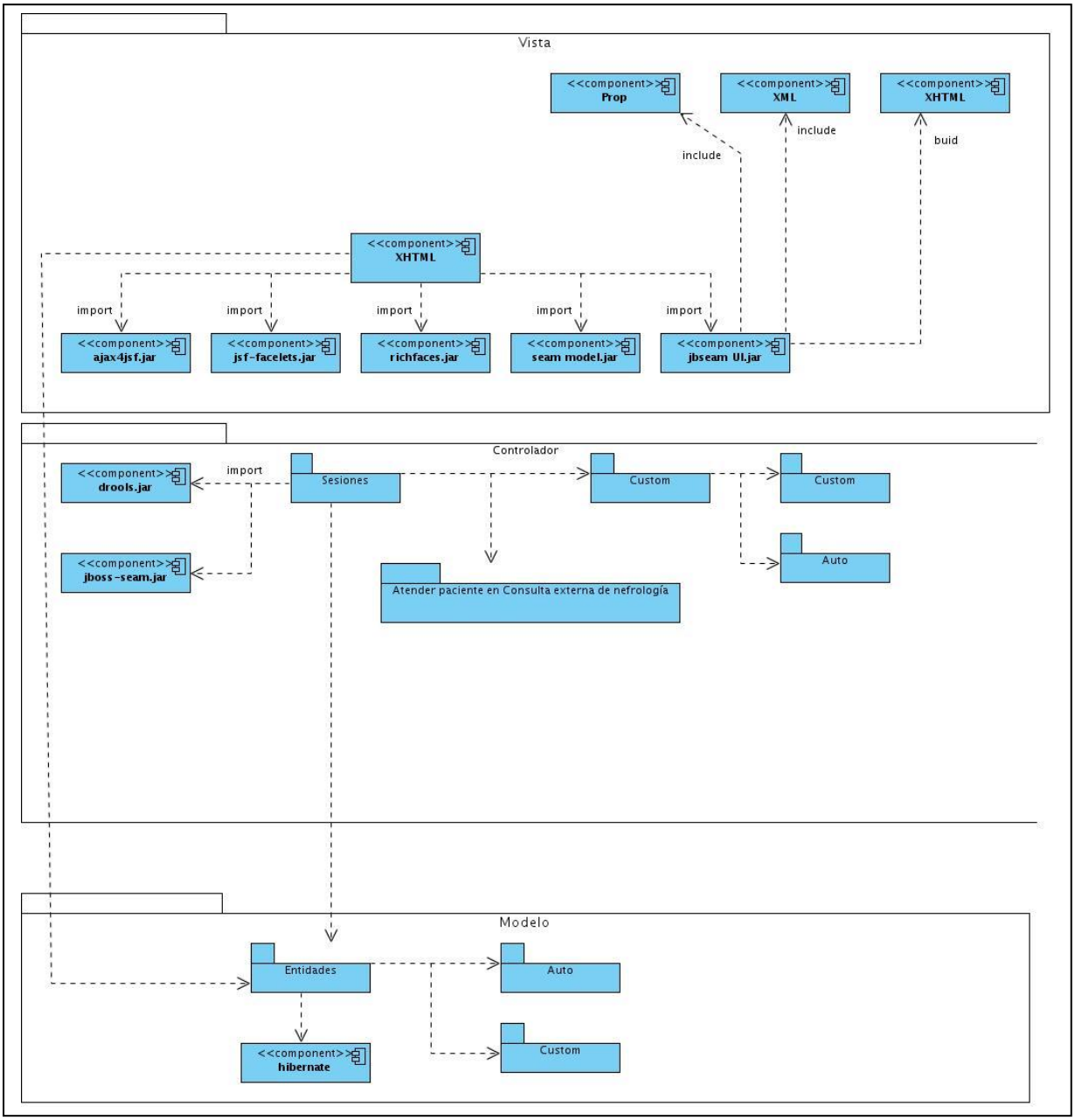


Figura 3.2 Diagrama de componentes

Conclusiones

En este capítulo se realizó una valoración del diseño propuesto por el analista de sistemas, donde se especifican las modificaciones que fue necesario realizar a este diseño, dado que no era óptimo para cubrir los requerimientos especificados.

A partir de los artefactos documentados, será más fácil comprender el alcance de la presente investigación y dar continuidad a la misma.

CAPÍTULO 4. PRUEBA.

El presente capítulo tiene como propósito documentar la validación de la presente investigación realizada durante la fase de pruebas, teniendo en cuenta los métodos de prueba empleados y los resultados obtenidos.

4.1 Prueba de caja negra.

Las pruebas de software, en inglés testing, son los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador o videojuego. Básicamente es una fase en el desarrollo de un sistema, consistente en probar las aplicaciones construidas. Las pruebas de una aplicación se integran dentro de las diferentes fases del ciclo de desarrollo del software. [40]

Las pruebas de caja negra se centran en lo que se espera de un sistema, es decir, intentan encontrar casos en que el mismo no se atiene a su especificación, por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar ocurriendo por dentro. Está especialmente indicada en aquellos componentes que van a tener interacción con el usuario y se apoyan en la especificación de requisitos. Los datos de prueba se escogerán atendiendo a las especificaciones del problema, a fin de verificar que el programa funcione adecuadamente.

Con este tipo de prueba se intentará 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.

Con los casos de prueba de caja negra se pretende demostrar además que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene. [41]

4.2 Métodos de prueba de caja negra

La Prueba de Caja Negra consta de varios métodos, entre los que se encuentran:

Métodos de prueba basados en grafos: en este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. En este método:

- Se crea un grafo de objetos importantes y sus relaciones.
- Se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores.

Partición equivalente: presenta la partición equivalente como un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos, de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Análisis de valores límite: los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límite. El análisis de valores límite es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, lleva la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, obtiene casos de prueba también para el campo de salida.

Adivinando el error: dado un programa particular, se conjetura, por la intuición y la experiencia, ciertos tipos probables de errores y entonces se escriben casos de prueba para exponer esos errores. Es difícil dar un procedimiento para esta técnica puesto que es en gran parte un proceso intuitivo. La idea básica es enumerar una lista de errores posibles o de situaciones propensas a error y después escribir los casos de prueba basados en la lista. [38]

Para validad la propuesta se utilizó el método Partición Equivalente.

4.3 Descripción de los casos de pruebas

Caso de prueba: Conjunto de condiciones o variables bajo las cuales el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. Se pueden realizar muchos casos de prueba para determinar que un requisito es completamente satisfactorio. Es una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, los resultados esperados y las condiciones bajo las que se debe probar.

Caso de prueba: Crear hoja nefrológica del donante

Escenarios	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Crear Hoja nefrológica del donante.	Se crea la Hoja nefrológica del donante correctamente con los datos correspondientes.	En la lista de donantes previamente seleccionados, se da clic en seleccionar y muestra la hoja nefrológica del donante a crear. Se introducen los datos.
EC 2: Existen campos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber datos obligatorios incompletos muestra un asterisco (*) rojo indicándolo.	En la lista de donantes previamente seleccionados, se da clic en seleccionar y muestra la hoja nefrológica del donante a crear. Se introducen los datos.
EC 3: Existen datos incorrectos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber datos inválidos muestra un asterisco (*) rojo indicándolo.	En la lista de donantes previamente seleccionados, se da clic en seleccionar y muestra la hoja nefrológica del donante a crear. Se introducen los datos.

Id del escenario	EC 1	EC 2	EC 3
Escenario	Crear Hoja nefrológica del donante exitosamente.	Existen campos incompletos	Existen datos incorrectos
Variable 1 (Peso)	V	V	V

Id del escenario	EC 1	EC 2	EC 3
Variable 2 (Frecuencia cardíaca)	V	I	V
Variable 3 (Sistólica)	V	V	I
Variable 4 (Diastólica)	V	V	I
Variable 5 (Observaciones)	V	V	V
Variable 6 (Conclusiones)	V	V	V
Respuesta del Sistema	El sistema crea la Hoja nefrológica del donante	El sistema no crea la Hoja nefrológica del donante.	El sistema no crea la Hoja nefrológica del donante.
Resultado de la Prueba	Exitosa	Exitosa	Exitosa

Caso de prueba: Crear hoja de trasplante renal

Escenarios	Descripción de la funcionalidad	Flujo Central
EC 1: Crear Hoja de trasplante renal.	Se crea la hoja de trasplante renal correctamente con los datos correspondientes.	En las opciones de la hoja ”Crear hoja de nefrología” se selecciona la opción “Crear hoja de trasplante renal” se seleccionan los datos y se da clic en el botón “Aceptar”.
EC 2: Existen campos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber datos obligatorios incompletos, no se crea la hoja.	En las opciones de la hoja ”Crear hoja de nefrología” se selecciona la opción “Crear hoja de trasplante renal” se seleccionan los datos y se da clic en el botón “Aceptar”.

Id del escenario	EC 1	EC 2
Escenario	Crear Hoja de trasplante renal exitosamente	Existen campos incompletos
Variable 1 (Estado aptitud)	V	V
Variable 2 (Tipo trasplante)	V	I
Variable 3 (Observaciones)	V	V
Respuesta del Sistema	El sistema crea la hoja de trasplante renal.	El sistema no crear la hoja de trasplante renal.
Resultado de la Prueba	Exitosa	Exitosa

Caso de prueba: Modificar hoja nefrológica del donante

Escenarios	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Modificar Hoja nefrológica del donante.	Se modifica correctamente la Hoja nefrológica del donante.	Una vez creada(s) la(s) hoja(s) se va al Consultar acciones realizadas de la Hoja de trasplante renal y se muestra la(s) hoja(s) creada(s), donde se muestran siguientes: peso y frecuencia cardiaca. Se da clic en la opción modificar. Se introducen los datos correctos, clic en el botón “Modificar”.
EC 2: Existen campos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber datos obligatorios incompletos muestra un asterisco (*) rojo indicándolo.	Una vez creada(s) la(s) hoja(s) se va al Consultar acciones realizadas de la Hoja de trasplante renal y se muestra la(s) hoja(s) creada(s), donde se muestran siguientes: peso y frecuencia cardiaca. Se da clic en la opción modificar.

Escenarios	Descripción de la funcionalidad	Flujo Central
		Se introducen los datos correctos, clic en el botón “Modificar”. Se da clic en la opción modificar. Se introducen los datos correctos, clic en el botón “Modificar”.
EC 3: Existen datos incorrectos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber datos inválidos muestra un asterisco (*) rojo indicándolo.	Una vez creada(s) la(s) hoja(s) se va al Consultar acciones realizadas de la Hoja de trasplante renal y se muestra la(s) hoja(s) creada(s), donde se muestran siguientes: peso y frecuencia cardiaca. Se da clic en la opción modificar. Se introducen los datos correctos, clic en el botón “Modificar”. Se da clic en la opción modificar. Se introducen los datos correctos, clic en el botón “Modificar”.

Id del escenario	EC 1	EC 2	EC 3
Escenario	Modificar Hoja nefrológica del donante exitosamente.	Existen campos incompletos	Existen datos incorrectos
Variable 3 (Peso)	V	V	V
Variable 4 (Frecuencia cardíaca)	V	I	V
Variable 3 (Sistólica)	V	V	I
Variable 4 (Diastólica)	V	V	I

Id del escenario	EC 1	EC 2	EC 3
Variable 5 (Observaciones)	V	V	V
Variable 6 (Conclusiones)	V	V	V
Respuesta del Sistema	El sistema modifica la Hoja nefrológica del donante	El sistema no modificar la Hoja nefrológica del donante.	El sistema no modificar la Hoja nefrológica del donante.
Resultado de la Prueba	Exitosa	Exitosa	Exitosa

Caso de prueba: Modificar hoja de trasplante renal

Escenarios	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Modificar Hoja de trasplante renal.	Se modifica la hoja de trasplante renal correctamente con los datos correspondientes	En la interfaz “Crear hoja de nefrología” se activa la opción “Consultar acciones realizadas”, se da clic y muestra datos de la hoja de trasplante creada. Clic en la opción “Modificar”. Se seleccionan los datos. Clic en el botón “Modificar”.
EC 2: Existen campos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber datos obligatorios incompletos, no se crea la hoja.	En la interfaz “Crear hoja de nefrología” se activa la opción “Consultar acciones realizadas”, se da clic y muestra datos de la hoja de trasplante creada. Clic en la opción “Modificar”. Se seleccionan los datos y se deja un campo vacío. Clic en el botón “Modificar”.

Id del escenario	EC 1	EC 2
Escenario	Se modifica la hoja de trasplante renal correctamente con los datos correspondientes	Existen campos incompletos
Variable 1 (Estado aptitud)	V	V
Variable 2 (Tipo trasplante)	V	I
Variable 3 (Observaciones)	V	V
Respuesta del Sistema	El sistema modifica la Hoja de trasplante renal.	El sistema no modificar la Hoja de trasplante renal
Resultado de la Prueba	Exitosa	Exitosa

Conclusiones

En este capítulo se realizó el modelo de prueba, donde se aplicó el método partición equivalente, obteniéndose los casos de prueba como artefactos generados del flujo de trabajo, los que permitieron validar el resultado de esta investigación.

CONCLUSIONES

Al culminar la presente investigación se han cumplido el objetivo y las tareas planteadas, obteniéndose los siguientes resultados:

- Se asimiló la arquitectura propuesta por el Departamento Gestión Hospitalaria para el desarrollo de sus aplicaciones, lo cual permitió la construcción de un componente robusto y flexible.
- La utilización de pautas para el proceso de desarrollo, garantizó uniformidad y homogeneidad en los artefactos obtenidos a partir de este.
- La ejecución de pruebas de caja negra garantizó la calidad necesaria para la futura explotación de las funcionalidades implementadas.
- Se obtuvo la Hoja de trasplante renal del componente Consulta Externa de Nefrología del Sistema de Información Hospitalaria alas HIS, que contribuirá a elevar la calidad de la atención nefrológica que se les brinda a los pacientes que requieren de trasplante renal.

RECOMENDACIONES

Por la experiencia obtenida en el desarrollo de la Hoja de trasplante renal del componente Consulta Externa de Nefrología, la autora recomienda para próximas versiones:

- Desarrollar un componente que gestione los procesos que se realizan con los pacientes de la Lista de espera nacional de receptores renales.

- Desarrollar un componente para la generación de alertas, relacionadas con el trasplante renal.

- Reimplementar la funcionalidad del sistema que se encarga de dar la propuesta de estado de aptitud de un paciente que se encuentra en espera de trasplante, teniendo en cuenta además de los elementos hasta hoy empleados, los estudios de Cistografía, Estudio Vascular y Ecocardiograma.

REFERENCIAS BIBLIOGRÁFICAS

1. Historia de la nefrología en Latinoamérica [En línea] martes 1 de febrero de 2011 http://nefrologia-urologia.blogspot.com/

2. Morales Torres, Leidis Marian y Avila Rojas, Edelberto. Diseño de la Consulta Externa de Nefrología y del proceso de Trasplante renal del Sistema de Información Hospitalaria alas HIS. Ciudad de La Habana. Cuba : s.n., junio 2010.

3. Rep. Dom. 15 de Junio de 111 Hora local 4:48 P.M.http://www.arqhys.com/casas/externa-consulta.html

4. MedLine Información de salud para usted [En línea] martes 11 de febrero de 2011http://www.nlm.nih.gov/medlineplus/spanish/ency/article/000471.htm

5. Renal info [En línea] martes 11 de abril de 2009 http://spain.renalinfo.com/opciones_de_tratamiento/trasplante_1.html

6. curlango@yaqui.mxl.uabc.mx. UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA. [En línea] 25 mayo de 2009 http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm.

7. González Cornejo, José Enrique. ¿Qué es UML? 2010-02-14. Página Web. Disponible en: http://www.docirs.cl/uml.htm

8. Ídem 7

9. Caro, Patricio Salinas y K., Nancy Histchfeld. http://www.dcc.uchile.cl/. [En línea] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

10. Cristhian Herrera [En línea] 15 noviembre 2008 http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/JSF#ventajasEinconvenientes

11. Juan Cordovez. fap-devel. [En línea] 13 de junio de 2009. http://code.google.com/p/fap-devel.

12. fap-devel. [En línea] 16 de junio de 2009. http://code.google.com/p/fap-devel/.

13. fap-devel. [En línea] 28 de junio de 2009. http://code.google.com/p/fap-devel/.

14. Garrett, Jesse James. Maestros del web. [En línea] 11 de Junio de 2005. [Citado el: 9 de Febrero de 2011.] http://www.maestrosdelweb.com/editorial/ajax/.

15. MADEJA/Ajax4JSF. [En línea] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/Ajax4JSF.

16. W3C World Wide Web. [En línea] 09 de enero de 2008. http://www.w3c.es/divulgacion/guiasbreves/hojasestilo.

17. W3C World Wide Web. [En línea] 07 de enero de 2008. http://www.w3c.es/divulgacion/guiasbreves/XHTML.

18. http://nahoul.byethost32.com. [En línea] 21 de octubre de 2010. http://nahoul.byethost32.com/doku.php?id=piw1_14.

19. Escrito por _Tes_ hace 2 años bajo una licencia de Creative Commons. http://es.debugmodeon.com. [En línea] http://es.debugmodeon.com/articulo/que-es-jboss-seam.

20. ¿Qué es JBoss Seam? 2010-2-20. Página Web. Disponible en: http://es.debugmodeon.com/articulo/que-es-jboss-seam

21. Crespo, Cesar. Introducción a Hibernate. 2010-02-17. Página Web. Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernate

22. Red Hat Middleware Relational Persistence for Java and .NET.2009 Disponible en: https://hibernate.bluemars.net/

23. Cristhian Herrera. http://www.adictosaltrabajo.com. [En línea] 17 de octubre de 2007. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=EJB3vsSpring.

24. Java Persistence API FAQ. Sun Microsystems. (s.f.). Recuperado el Febrero de 2010, de http://java.sun.com/javaee/overview/faq/persistence.jsp

25. [En línea] 10 de marzo de 2009. http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/ejb/sesion01-apuntes.htm

26. JBoss jBPM. 2010-02-18. Página Web. Disponible en: http://tratandodeentenderlo.blogspot.com/2009/06/jboss-jbpm.html

27. Herrera, Cristhian. Adictos al trabajo. [En línea] [Citado el: 9 de Febrero de 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=EJB3vsSpring#1.2.1.Enterprise%20JavaBeans|outline.

28. PostgreSQL Maestro 8.3. Marzo 2008. Disponible en: http://www.freedownloadscenter.com/es/Programacion/Base_de_Datos_y_Redes/PostgreSQL_Maestro.html .

29. Guía ubuntu. febrero 2010 disponible en: http://www.guia-ubuntu.org/index.php?title=PostgreSQL

30. Rondon Grados, Luis. JPA - Java Persistence API. 2010-02-19. Página Web. Disponible en: http://luchorondon.blogspot.com/2009/04/jpa-java-persistence-api.html

31. Descripción de Eclipse SDK 3.3.2. 2010-02-13. Página Web. Disponible en: http://www.boxsoftware.net/programas/eclipse-sdk-3-3-2.asp

32. Descripción de Eclipse SDK 3.3.2. 2010-02-13. Página Web. Disponible en: https://www.ohloh.net/p/pgadmin

33. Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME)) 6.0. 2010-02-19. Página Web. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/

34. PgAmin para aprender [En linea] 30 marzo 2009 http://postgresql.org.pe/articles/postgresql.pdf

35. [En línea] miércoles 22 julio de 2009 http://arleytriana.blogspot.com/2009/07/implementacion-del-patron-clasico-de.html

36. Idem 35

37. Publicado por Gustavo [En línea] miércoles 30 julio de 2009 http://cup-coffe.blogspot.com/2010/02/introduccion-al-aspnet-mvc.html

38. [En línea] miércoles 3 noviembre de 2009 http://sites.google.com/site/jalexiscv/modelosdedatos

39. [En línea] 10 de marzo de 2009. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/moreno_a_jl/capitulo5.pdf

40. [En línea] jueves, 13 de octubre de 2005. http://pruebasoftware.blogcindario.com/2005/10/00002-disenos-de-casos- de-prueba.html

41. [En línea] jueves, 13 de diciembre de 2009. http://indalog.ual.es/mtorres/LP/Prueba.pdf

BIBLIOGRAFÍA

1. Morales, Leidis Marian Torres Avila y Edelberto Rojas**.** *Diseño de la Consulta Externa de Nefrología y del proceso de Trasplante renal del Sistema de Información Hospitalaria alas HIS.* Ciudad de La Habana. Cuba : s.n., junio 2010.

2. curlango@yaqui.mxl.uabc.mx**.** UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA. [En línea] http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm.

3. Caro, Patricio Salinas y K., Nancy Histchfeld. http://www.dcc.uchile.cl/. [En línea] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

4. Juan Cordovez**.** fap-devel. [En línea] 13 de junio de 2009. http://code.google.com/p/fap-devel.

5. fap-devel. [En línea] 16 de junio de 2009. http://code.google.com/p/fap-devel/.

6. *fap-devel.* [En línea] 28 de junio de 2009. http://code.google.com/p/fap-devel/.

7. MADEJA/Ajax4JSF. [En línea] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/Ajax4JSF.

8. W3C World Wide Web. [En línea] 09 de enero de 2008. http://www.w3c.es/divulgacion/guiasbreves/hojasestilo.

9. W3C World Wide Web. [En línea] 07 de enero de 2008. http://www.w3c.es/divulgacion/guiasbreves/XHTML.

10. [En línea] 21 de octubre de 2010. http://nahoul.byethost32.com/doku.php?id=piw1_14. http://nahoul.byethost32.com.

11. Escrito por _Tes_ hace 2 años bajo una licencia de Creative Commons**.** [En línea] http://es.debugmodeon.com/articulo/que-es-jboss-seam. http://es.debugmodeon.com.

12. Herrera, Cristhian**.** [En línea] 17 de octubre de 2007. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=EJB3vsSpring. http://www.adictosaltrabajo.com.

13. [En línea] http://descargar.mp3.es/lv/group/view/kl38900/Java_Runtime_Environment_JRE_.htm. http://descargar.mp3.es.

14 [En línea] 09 de diciembre de 2010. http://parasitovirtual.wordpress.com/2010/12/09/el-salto-de-java-se-a-java-ee/. http://parasitovirtual.wordpress.com.

15. Argulo, Ivan**.** [En línea] 20 de enero de 2009. http://ivanargulo.wordpress.com/2009/01/20/modelo-vista-controlador/. http://ivanargulo.wordpress.com.

16. http://www.google.com.cu/imgres?imgurl=http://1.bp.blogspot.com. [En línea] 18 junio del 2008 http://www.google.com.cu/imgres?imgurl=http://1.bp.blogspot.com/_QOEod-XOIJ0/S7n5y95m_UI/AAAAAAAAAAg/awRWtsjMwnM/s1600/rup_model.jpg&imgrefurl=http://wwwderian.blogspot.com/&usg=__hwdGY-iolsZ_RBM31hdRq8_ETgg=&h=525&w=700&sz=55&hl=es&start=4&zoom=1&um=1&it.

17. http://es.scribd.com. [En línea] 10 de enero de 2009. http://es.scribd.com/doc/28551849/Manual-Jsf.

18. http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework/. [En línea] 10 de enero de 2009

19. Díaz Rodríguez, Implementación de los procesos del movimiento hospitalario del Sistema de Información Hospitalaria alas HIS. Ciudad de La Habana. Cuba : s.n., junio 2010.

20. Petrolito, Dr. José. Historia de la Asociación. [En línea] 03 de 2001. [Citado el: 05 de 12 de 2009.] http://www.renal.org.ar/revista/53/5303.htm.

21. Visual Limes**,** S.L**.** Nefrosoft®HD. [En línea] [Citado el: 06 de 04 de 2011.] http://www.nefrosoft.com/.

22. UCI**.** eva.uci.cu. [En línea] [Citado el: 09 de 04 de 2011.] http://eva.uci.cu/mod/resource/view.php?id=21621&subdir=/El_lenguaje_unificado_de_modelado._Manual_de_Referencia.

23. Informática Profesional. [En línea] 2003. [Citado el: 09 de 04 de 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Ajax4Jsf.

24. Cordobo**.** Desarrollo en Web. [En línea] [Citado el: 09 de 04 de 2011.] http://blogs.antartec.com/desarrolloweb/2008/12/facelets-y-jsf-uso-de-templates/.

25. Web Taller. [En línea] 2003-2008. [Citado el: 20 de 04 de 2010.] http://www.webtaller.com/construccion/lenguajes/html/lecciones/que-es-xhtml.php.

26. librosweb.es. [En línea] [Citado el: 19 de 04 de 2010.] http://librosweb.es/css/capitulo1.html.

27. wilmanchamba**.** Desarrollo de aplicaciones web bajo entornos Java, frameworks de desarrollo J2EE. [En línea] 20 de 02 de 2008. [Citado el: 09 de 04 de 2010.] http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework/.

28. jabaHispano**,** Asociación**.** javaHispano. [En línea] 2002-2007. [Citado el: 10 de 02 de 2010.] http://www.javahispano.org/contenidos/es/liberado_drools_3_0/.

29. Hibernate. [En línea] [Citado el: 10 de 02 de 2010.] http://es.answers.yahoo.com/question/index?qid=20081014162041AAkTKNy.

30. Informática Profesional 1. [En línea] 2003. [Citado el: 10 de 05 de 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernateVSEJB3.

31. El Holgazán. [En línea] [Citado el: 10 de 05 de 2010.] http://www.elholgazan.com/2007/08/jpa-java-persistence-api.html.

32. Lenguajes de Programación. [En línea] 2009. [Citado el: 15 de 04 de 2011.] http://www.lenguajes-de-programacion.com/programacion-java.shtml.

33. Codazzi, Instituto Geográfico Agustín**.** SGBD. [En línea] 2004. [Citado el: 10 de 02 de 2011.] http://www.igac.gov.co:8080/igac_web/UserFiles/File/ciaf/TutorialSIG_2005_26_02/paginas/ctr_sistemasdegestiondebasededatos.htm.

34. [Online] [Cited: 02 05, 2010.] http://www.netpecos.org/docs/mysql_postgres/x108.html.

35 . Visual Paradigm. [En línea] [Citado el: 06 de 12 de 2010.] http://www.visual-paradigm.com/product/vpuml.

36 Ciencia.net. [En línea] [Citado el: 27 de 12 de 2011.] www.ciencia.net/VerArticulo/Riñón-artificial.

37 . [En línea 2009. [Citado el: 15 de 04 de 2011.] http://www.ambysoft.com/essays/flootSpanish.html

38. http://descargar.mp3.es. [En línea] http://descargar.mp3.es/lv/group/view/kl38900/Java_Runtime_Environment_JRE_.htm.

39. http://parasitovirtual.wordpress.com. [En línea] 09 de diciembre de 2010. http://parasitovirtual.wordpress.com/2010/12/09/el-salto-de-java-se-a-java-ee/.

40. Ivan Argulo**.** *http://ivanargulo.wordpress.com.* [En línea] 20 de enero de 2009. http://ivanargulo.wordpress.com/2009/01/20/modelo-vista-controlador/.

41. *http://www.google.com.cu/imgres?imgurl=http://1.bp.blogspot.com.* [En línea] 20 de enero de 2009 http://www.google.com.cu/imgres?imgurl=http://1.bp.blogspot.com/_QOEod-XOIJ0/S7n5y95m_UI/AAAAAAAAAAg/awRWtsjMwnM/s1600/rup_model.jpg&imgrefurl=http://wwwderian.blogspot.com/&usg=__hwdGY-iolsZ_RBM31hdRq8_ETgg=&h=525&w=700&sz=55&hl=es&start=4&zoom=1&um=1&it.

42. [En línea] 10 de enero de 2009. http://es.scribd.com/doc/28551849/Manual-Jsf.

43. [En línea] 10 de marzo de 2009. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/moreno_a_jl/capitulo5.pdf

44. [En línea] jueves, 13 de octubre de 2005. http://pruebasoftware.blogcindario.com/2005/10/00002-disenos-de-casos-de-prueba.html

45. [En línea] jueves, 13 de diciembre de 2009. http://indalog.ual.es/mtorres/LP/Prueba.pdf

46. Historia de la nefrología en Latinoamérica [En línea] martes 1 de febrero de 2011 http://nefrologia-urologia.blogspot.com/

47. Rep. Dom. 15 de Junio de 111 Hora local 4:48 P.M.http://www.arqhys.com/casas/externa-consulta.html

48. Renal info [En línea] martes 11 de abril de 2009 http://spain.renalinfo.com/opciones_de_tratamiento/trasplante_1.html

49. González Cornejo, José Enrique. ¿Qué es UML? 2010-02-14. Página Web. Disponible en: http://www.docirs.cl/uml.htm

50. Ídem 49

51. ¿Qué es JBoss Seam? 2010-2-20. Página Web. Disponible en: http://es.debugmodeon.com/articulo/que-es-jboss-seam

52. Crespo, Cesar. Introducción a Hibernate. 2010-02-17. Página Web. Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernate

53. Red Hat Middleware Relational Persistence for Java and .NET.2009 Disponible en: https://hibernate.bluemars.net/

54. Java Persistence API FAQ. Sun Microsystems. (s.f.). Recuperado el Febrero de 2010, de http://java.sun.com/javaee/overview/faq/persistence.jsp

55. [En línea] 10 de marzo de 2009. http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/ejb/sesion01- apuntes.htm

56. JBoss jBPM. 2010-02-18. Página Web. Disponible en: http://tratandodeentenderlo.blogspot.com/2009/06/jboss-jbpm.html

57. Herrera, Cristhian. Adictos al trabajo. [En línea] [Citado el: 9 de Febrero de 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=EJB3vsSpring#1.2.1.Enterprise%20JavaBeans|outline.

58. PostgreSQL Maestro 8.3. Marzo 2008. Disponible en: http://www.freedownloadscenter.com/es/Programacion/Base_de_Datos_y_Redes/PostgreSQL_Maestro.html .

59. Guía ubuntu. febrero 2010 disponible en: http://www.guia-ubuntu.org/index.php?title=PostgreSQL

60. Rondon Grados, Luis. JPA - Java Persistence API. 2010-02-19. Página Web. Disponible en: http://luchorondon.blogspot.com/2009/04/jpa-java-persistence-api.html

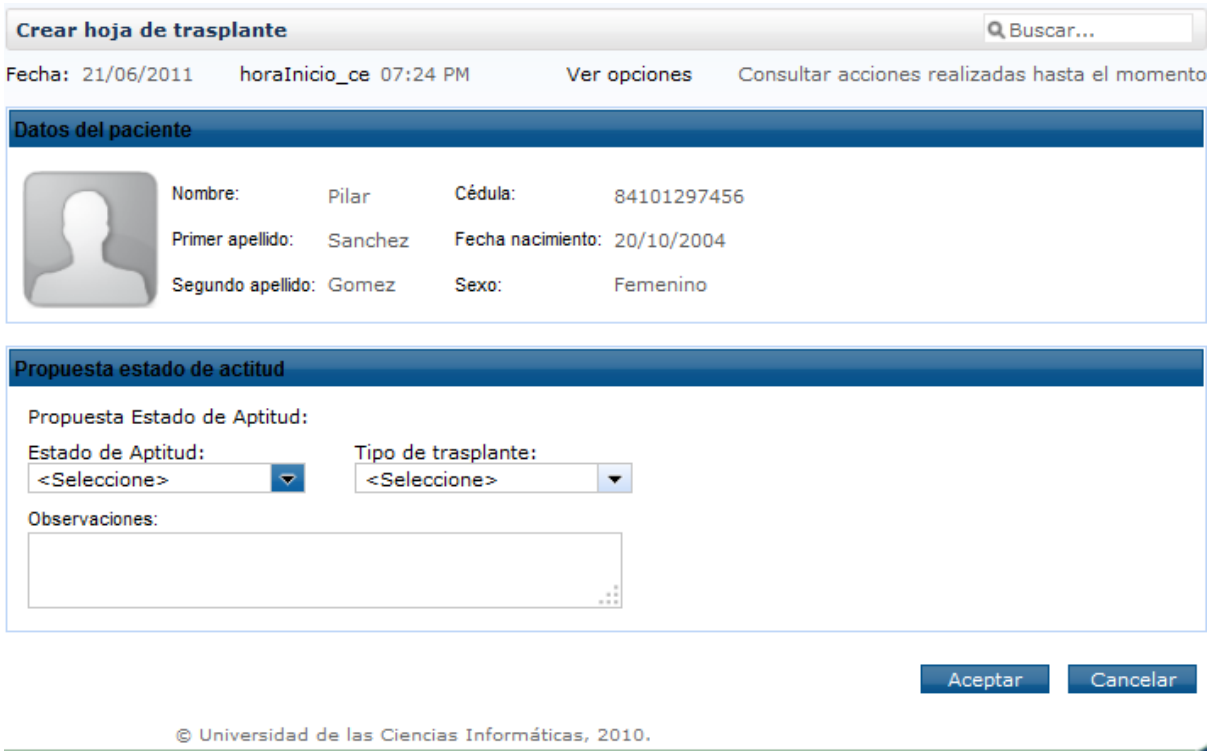
61. Descripción de Eclipse SDK 3.3.2. 2010-02-13. Página Web. Disponible en: http://www.boxsoftware.net/programas/eclipse-sdk-3-3-2.asp

62. Descripción de Eclipse SDK 3.3.2. 2010-02-13. Página Web. Disponible en: https://www.ohloh.net/p/pgadmin
63. Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME)) 6.0. 2010-02-19. Página Web. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/
64. PgAmin para aprender [En linea] 30 marzo 2009 http://postgresql.org.pe/articles/postgresql.pdf
65. [En línea] miércoles 22 julio de 2009 http://arleytriana.blogspot.com/2009/07/implementacion-del-patron-clasico-de.html
66. Idem 65

67. Publicado por Gustavo [En línea] miércoles 30 julio de 2009 http://cup-coffe.blogspot.com/2010/02/introduccion-al-aspnet-mvc.html

68. [En línea] miércoles 3 noviembre de 2009 http://sites.google.com/site/jalexiscv/modelosdedatos

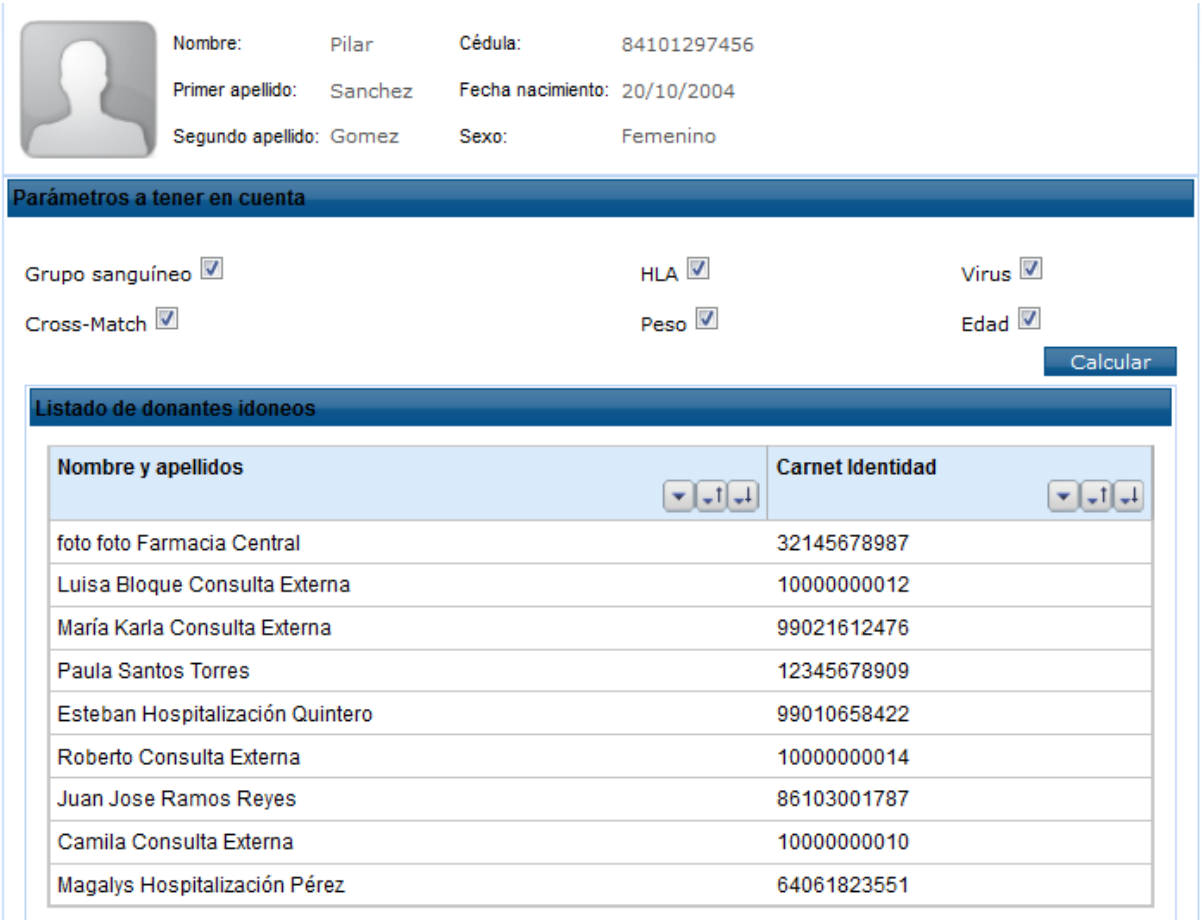
ANEXOS



Anexo 1. Interfaz: Crear hoja de trasplante renal.

Anexo 2. Interfaz: Resumen de estudios para estado de aptitud.

Anexo 3. Interfaz: Crear hoja nefrológica del donante.



Anexo 4. Interfaz: Determinar donante idóneo.

GLOSARIO

- **Aplicación o Sistema Informático:** Programas con los cuales el usuario final interactúa a través de una interfaz y que realizan tareas útiles para éste.

- **Framework:** Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otros software para ayudar a desarrollar y a unir los diferentes componentes de un proyecto.

- **Informática:** Disciplina que estudia el tratamiento automático de la información utilizando dispositivos electrónicos y sistemas computacionales.

- **Software:** Conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema.

- **Servidor de aplicaciones:** Es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de Internet y utilizando el protocolo HTTP.