Universidad de las Ciencias Informáticas

Facultad # 7

Título: Implementación del Sistema de Información de Bancos de Sangre de los Hospitales

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autores: Keila Torres Santiesteban
Eduardo Cuello Martínez

Tutor: Ángel Fabra Torres

Consultante: Raiza Rodríguez Carcassés

Asesor: Juana Isabel Pérez

Ciudad Habana, Julio 2007

¨Si al franquear una montaña en la dirección de una estrella, el viajero se deja absorber demasiado por los problemas de la escalada, se arriesga a olvidar cual es la estrella que lo guía.¨

Antoine de Saint-Exupery

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los 6 días del mes de julio del año 2007

Keila Torres Santiesteban **Eduardo Cuello Martínez**

_____ _____

Firma del Autor Firma del Autor

Ángel Fabra Torres

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor: Lic. Angel Fabra Torres
Graduado de Lic. en Ciencias de la Computación por la Universidad de Oriente, año 2004.
Actualmente labora como profesor en la Universidad de las Ciencias Informáticas.
Posee la categoría docente de Instructor.
e-mail: afabra@uci.cu

Asesora: Juana Isabel Pérez Rodríguez
Licenciada en Lengua Inglesa. Especialidad: Educación
Categoría docente: Instructora
e-mail: Jane@uci.cu

Consultante: Lic. Raiza Rodriqguez Carcasses.
Actualmente labora en la Dirección de Investigaciones Agropecuarias
Centro de Ingeniería Genética y Biotecnología (CIGB)
e-mail: raisa.rodriguez@cigb.edu.cu

AGRADECIMIENTOS

Keila……..

A mi madre por estar siempre brindándome todo cuanto he necesitado, por quererme como lo ha hecho, por cuidarme tanto, por tanta entrega, por estar siempre conmigo, por todo y por tantas cosas…….

A mi padre porque ha sabido estar siempre cerca, por ser el mejor padre del mundo, por quererme y tenerme siempre presente…

A mi abuela por ser esa estrella que siempre está no solo para verla sino para creer en ella e inspirarse en cada momento, por adorarme, por tantos años de preocupación, dedicación y entrega….

A mi hermanita que tan linda ha sido conmigo todos estos años, por tanto cariño y por estar siempre apoyándome, ayudándome, cuidándome……

A mi novio por estar tan cerca de mí ayudándome, comprendiéndome, queriéndome…

A mis tíos Ramón, Luis Ángel y Raiza por quererme, ayudarme y estar para mí, siempre que los necesito….

A mi abuelo Use por preocuparse, por quererme y creer tanto en mi…

A Juli, La China, Silantra, por ese apoyo que me dieron y me dan, que tan importante ha sido para mi…

Al Clan por la amistad que me brindaron, por estar cuando las necesité, ayudándome y comprendiéndome……

A nuestro Comandante Fidel, por idear este proyecto, por crear una sociedad tan justa y por creer tanto en esta juventud que le debe todo cuanto somos….

A todos los profesores que me ayudaron en mi formación profesional y en la confección de esta tesis…

Eduardo

A mi mamá por apoyarme durante toda mi vida como estudiante.

A mi papá por apoyarme y aconsejarme en cada momento que lo necesité.

A mi abuelita linda por quererme tanto.

A mi novia por estar a mi lado y apoyarme en todos estos años.

A mis tíos Pepi, Leonor y Negri, por alentarme siempre.

A todos mis amigos por estar siempre presentes.

A mi tutor y a todos los que contribuyeron con esta investigación.

Al comandante Fidel Castro y a la revolución por crear esta maravillosa escuela y darnos así la oportunidad de formarnos en ella.

A todos y cada uno. Muchas Gracias

DEDICATORIA

Keila

A mis padres que hicieron posible no solo que viviera sino que creciera y me formara como lo he hecho, que tan orgullosos y felices se sienten, por la obtención de este resultado…
A mi abuelita que merece este título y todos los logros que yo pueda tener…
A mi abuelo Chicho que estaría muy orgulloso de mí si estuviera entre nosotros…
A toda mi familia que con tanto amor ayudaron a que me convirtiera en esta persona que soy hoy y que con su educación y apoyo contribuyeron a que estos cinco años tuvieran un fin tan satisfactorio.
A la Revolución por haberme permitido la realización de este sueño.
A todas aquellas personas, que un día nos hicieron comprender que no existe fuerza que pueda simplificar el valor de un hombre, cuando es fruto de entrega a otras personas que compactando sentimientos de humanidad y valores muchos más altos, buscan dentro de nuestros cuerpos el camino de la victoria, el hombre íntegro a la sociedad.

Eduardo

A mis padres, por educarme y apoyarme en todos los momentos.
A mi abuela querida, que durante todo este tiempo ha estado apoyándome en cada momento.
A mi novia, por apoyarme durante todos estos años de estudios.
A mis hermanos.
A mis tíos, que sé se sienten muy orgullosos de mí y me han dado aliento en cada paso.
En general, a toda mi familia, ya que todos ellos desearon mucho que yo alcanzara este titulo y los hace tan felices como a mi.
A mis amigos y amigas.
A todos aquellos que de una forma u otra contribuyeron en mi formación como estudiante, como profesional revolucionario y como persona.

RESUMEN

Los Bancos de Sangre como parte del Sistema Nacional de Salud (SNS) están inmersos en el proceso de informatización, con este objetivo y para automatizar los procesos que hoy resultan trabajosos y que provocan un difícil manejo por parte de médicos y otro personal vinculado a esta labor. El objetivo de este trabajo es llevar a cabo la Implementación de un sistema informático para la gestión de los procesos de información hospitalaria en los bancos de sangre de los hospitales.

Para el logro del mismo se empleó como tecnología, ASP.NET y herramienta de desarrollo Visual Studio 2005, como lenguaje de programación del lado del servidor C#. Además se utilizó AJAX con el objetivo de agilizar la respuesta del sistema en cada uno de los procesos y como proveedor de base de datos NPGSQL.

Con la implementación del sistema, se espera obtener como resultados: la creación de una aplicación Web que agilice el trabajo del personal que se encuentre vinculado al Departamento de Banco de Sangre; la obtención de informes estadísticos rápidos, de las donaciones hechas mensual y anualmente con los datos que se quieran en los mismos así como; el logro que los datos que son archivados en la base de datos a través de la aplicación se puedan obtener en las próximas donaciones y su obtención resulte fácil.

Palabras Claves: Sistemas, Implementación, Bancos de Sangre.

TABLA DE CONTENIDO

LISTADO DE TABLAS

INTRODUCCIÓN

Se considera que la Medicina es prácticamente contemporánea con la humanidad. Desde la Edad Antigua, el hombre trató de mantener la salud, con prácticas rudimentarias que se fueron aprendiendo de generación en generación, acumulándose un gran caudal de conocimientos médicos.

Junto a la Medicina y otras ciencias, el hombre fue manejando un elemento vital para el enriquecimiento y desarrollo de todas las ciencias: "la información". Estos procesos de acumulación y asimilación de la información, desde el punto de vista filosófico, han contribuido al desarrollo cultural de la humanidad, al desarrollo de las fuerzas productivas, y en general, a alcanzar estadíos superiores en el desarrollo económico, social y científico de la civilización humana.

Desde los primeros momentos del triunfo de la Revolución, y como uno de los puntos a cumplir del programa del Moncada, el desarrollo de la salud pública constituye un elemento que distinguiría el proceso revolucionario. Innumerables han sido los logros alcanzados por la medicina cubana en estos años de Revolución y los esfuerzos realizados por el Estado Socialista para mantener una atención sanitaria a la altura de países desarrollados. Entre ellos: la modernización del Sistema Nacional de Salud (S.N.S), la construcción de modernos hospitales, como el Hospital "Hermanos Ameijeiras", entre otros que han permitido la introducción de tecnologías de punta para servir de apoyo a la asistencia médica.

Con el actual desarrollo de las tecnologías han ido surgiendo un conjunto de herramientas que puestas al servicio de empresas y entidades ayuda a lograr un mejor desarrollo de los procesos y una mejor eficiencia de los mismos.

La medicina en el país obtiene día a día notables avances tanto científicos como tecnológicos y muchos de estos han sido alcanzados gracias a la aplicación de herramientas informáticas en muchos de sus procesos.

Los hospitales son instituciones de salud que presentan un entorno muy complejo en cuanto a la gestión y procesamiento de la información. Debido a la gran cantidad de especialidades y servicios que se brindan en los ellos, los mismos se encuentran divididos por áreas o departamentos de especialidades, para una mejor organización y enfoque del hospital como entidad. Esto ha traído consigo que los HIS (Hospital Information System) desarrollados en el mundo estén divididos por módulos correspondientes a cada una

de las áreas del hospital, éstos se encuentran estrechamente relacionados y responden generalmente a un producto único, que no es más que la integración de todos los módulos, aunque algunas soluciones permiten que los módulos funcionen de manera independiente.

Una de las áreas de mayor importancia en los hospitales, es el Banco de Sangre. En esta se maneja todo lo relacionado con las donaciones de sangre, donantes, transfusiones, pedidos de sangre y/o sus componentes, reportes estadísticos etc.

El Banco de Sangre es la unidad ejecutiva del Instituto Nacional de Cardiología, responsable de la disposición de productos sanguíneos con oportunidad y en óptimas condiciones, para la realización de los diferentes procedimientos médicos que se les prescriben a los pacientes en los servicios asistenciales; por lo que es un servicio vital para la comunidad.

Es un Centro Sanitario cuyas tareas fundamentales son:

- Extracción de sangre o de alguno de sus componentes.
- Análisis, fraccionamiento y conservación de la sangre y derivados.
- Distribución a todos los centros hospitalarios y clínicas.

En el se llevan a cabo diversos procesos y estudios para garantizar que la sangre que se le transfunde a los pacientes cumpla todos los requisitos de calidad que exige la Legislación Sanitaria vigente, así como para garantizar que la sangre del paciente y del donador, sean compatibles.

Para la realización de este trabajo, se investigó, se hicieron entrevistas en los departamentos del banco de sangre de diferentes hospitales, indagando la necesidad de cada trabajador y de cada paciente, para poder llegar a la conclusión de que es necesario mejorar de la forma más avanzada posible, el método que se utiliza actualmente. Además está la referencia bibliográfica, como fuente de información, donde se puede ver todo lo relacionado al tema en otros software ya implementados en diferentes países.

En la actualidad, Cuba es país bloqueado económicamente; pero no ha mermado en su empeño de desarrollar el campo de la medicina que es para esta nación una de las esferas más importantes y a la que se le dedican todos lo recursos necesarios para sus logros tan satisfactorios, no solo para la población cubana sino mundial pues se cuenta con ese espíritu solidario que ayuda a trabajar para el país y para todas las poblaciones que lo necesiten.

Hoy en día la informática cubana se va caracterizando por una mayor asimilación de las aplicaciones relacionadas con el OPEN SOURCE para deshacerse de las patentes de Microsoft® y lograr un mayor desarrollo interno en el campo de la producción de software. Por todo esto resulta importante y necesaria la informatización de los hospitales cubanos en este caso los bancos de sangre, con el objetivo de contribuir al avance tecnológico que se ha venido desarrollando.

Situación Problémica:

En la actualidad, uno de los problemas que se presenta en el banco de sangre es el manejo del donante con donaciones anteriores. Tradicionalmente, se han empleado tarjeteros que son utilizados en la admisión y que deben permitir a la recepcionista reunir algunos elementos orientadores sobre la conveniencia o no del ingreso al banco de sangre de tales personas.

Estos tarjeteros son difíciles de manipular, de actualizar y no cuentan con toda la información necesaria. Adicionalmente, por razones bioéticas, no contienen los resultados de las pesquisas realizadas a la sangre para la determinación de gérmenes que se trasmiten a través de las transfusiones, ya que ésta es considerada una información confidencial: sífilis (VDRL), virus de la inmunodeficiencia humana (VIH), hepatitis viral tipo B (HVB), hepatitis viral tipo C (HVC).

Un aspecto de gran importancia, en el momento del ingreso del futuro donante al banco de sangre, es la fecha de la última donación. Aunque existen amplias variaciones en el mundo en relación con el intervalo mínimo entre donaciones de sangre, en Cuba se ha establecido como medida de protección a la salud del individuo, que deben haber transcurrido más de 3 meses de la fecha de la última donación para admitir al donante.

Sorprendentemente, muchos olvidan su reciente donación o incluso la niegan, lo cual ha sido comprobado por investigaciones realizadas. También es conveniente conocer los antecedentes patológicos o de carácter epidemiológico que pueden haber sido diagnosticados en el interrogatorio o en el examen médico practicado en una donación anterior. Estos datos se recogen normalmente en la historia clínica y se archivan, y su consulta resulta difícil al momento de la nueva donación.

Finalmente, se desea conocer si el donante tuvo algún percance en su donación anterior, tal es el caso de una reacción vagal severa, hipersensibilidad local o los que tienen un sistema venoso de difícil venipuntura, lo que le permitiría al médico tomar una decisión al respecto.

También, se hace trabajoso el proceso de interacción con otros departamentos del hospital como son: el Bloque Quirúrgico, específicamente en el momento que se hacen peticiones de sangre al banco para realizar una intervención quirúrgica. Esto lleva consigo un conjunto de operaciones relacionadas como: verificar existencia/disponibilidad de sangre del tipo requerido en el banco de sangre, realizar las pruebas de compatibilidad entre la sangre del paciente y la del donante, entre otras.

Por todo lo dicho anteriormente se hace necesaria la creación de una aplicación para el fácil manejo de los datos de todas las personas que donan, de la sangre donada y para el trabajo de la misma para las futuras transfusiones, entre otras muchas cosas que se demostrarán con el desarrollo de este trabajo.

Problema a resolver:

¿Cómo automatizar la gestión de la información en los bancos de sangre de los hospitales?

Objeto de Estudio:

Procesos hospitalarios automatizados en los hospitales.

Campo de acción:

Procesos de información hospitalaria automatizados en los bancos de sangre de los hospitales.

Objetivos Generales:

Implementar un sistema informático para la gestión de los procesos de información hospitalaria en los bancos de sangre de los hospitales.

Objetivos Específicos:

- Desarrollar una aplicación Web que permita la gestión de la información en los Bancos de Sangre.
- Implementar una interfaz orientada al usuario que sea de fácil manejo para los mismos.
- Aplicar los patrones de implementación que sean más óptimos.

Tareas por estudiantes:

- Obtener Modelo de Diseño propuesto por el analista.
- Hacer un análisis crítico de la tecnología o lenguaje a utilizar.
- Analizar la Integración con otros componentes o partes del sistema. Describir los elementos que resultan interfaz de comunicación con el resto de las partes del sistema.
- Realizar las pruebas necesarias para garantizar el correcto funcionamiento de la aplicación implementada.
- Realizar la implementación utilizando los patrones de diseño establecidos en el Análisis.
- Brindar una interfaz gráfica orientada al usuario y a las exigencias del mercado internacional.
- Garantizar la seguridad de los datos.
- Garantizar la interoperabilidad y flexibilidad del sistema.

El documento quedó estructurado en tres capítulos, donde se desarrolla todo lo relacionado con el análisis de los procesos que se realizan en los Bancos de Sangre, así como la implementación de una aplicación que da solución a la problemática existente en este departamento y la validación de la misma.

En el Capítulo 1 se definieron los principales conceptos y procesos con los que se trabaja en los Bancos de Sangre, así como una sencilla descripción de los sistemas que a nivel mundial han sido implementados para este módulo, y las tecnologías y herramientas que a través de la investigación se obtuvieron.

En el Capítulo 2, se elaboró una crítica al diseño propuesto por el analista, dada la importancia del mismo para la implementación, se ofrece información sobre los módulos o componentes que son reusados en el desarrollo del sistema. Se describe el algoritmo más complejo con el que cuenta la aplicación explicando además la estructura de datos usada en el mismo, se describen las clases que fueron definidas, que sirven de documentación a los que den continuidad a la implementación del software y como se integran las mismas.

Y por último, en el Capítulo 3 se valida la solución propuesta a través de la realización de pruebas, el mismo se encuentra dividido en dos partes la explicación detallada de las pruebas de caja blanca que fueron realizadas al código del software y las pruebas de caja negra que se realizaron a la interfaz del mismo.

CAPITULO 1: FUNDAMENTACIÓN TEÓRICA

Los avances tecnológicos con los que se conviven en la actualidad y ante la necesidad de crear aplicaciones que permitan lograr una mayor calidad en el trabajo que hoy enfrenta en disímiles esferas de la sociedad, se hace necesaria la creación de software. Mediante la implementación de los mismos se debe ser capaz de resolver todos los problemas planteados.

Para lograr este objetivo, es necesaria la realización de un profundo análisis del negocio donde se desarrollan los procesos a automatizar, la investigación de sistemas que han sido implementados y el estudio de las principales tecnologías y herramientas que serán usadas para la implementación del módulo.

Este primer capítulo quedó integrado de la siguiente manera:

Para la comprensión del negocio se hace un estudio de los principales conceptos que se encuentran estrechamente relacionados a el. [Epígrafe 1.1]

Es preciso, además describir como se desarrollan los procesos fundamentales que se llevan a cabo en el banco de sangre, pues de este modo se garantiza que una vez automatizados el trabajo con los mismos no difieran en mucho al que hoy se lleva manualmente. [Epígrafe 1.2]

No solo en Cuba, sino en el mundo se han desarrollado software para el trabajo en los Bancos de Sangre, cuyo estudio ha servido para tener en cuenta las ventajas y defectos que tienen, los que son necesarios tener en cuenta para que la propuesta resulte la más óptima. [Epígrafe 1.4]

Existe una gran competencia entre los programas para desarrollar aplicaciones. A menudo surgen dudas sobre cuál usar, debido a que actualmente aparecen nuevas herramientas con mejorías, que se van imponiendo para el desarrollo de software. Es por esto, que surge la necesidad de hacer un estudio de las herramientas a utilizar, buscando obtener la información más actual de las mismos. [Epígrafe 1.5]

1.1 Conceptos básicos asociados al dominio del problema

Los conceptos que se encuentran a continuación están asociados al dominio del problema y resultan necesarios describir para lograr un mayor entendimiento del mismo.

1.1.1 Aféresis

La aféresis es la técnica mediante la cual se separan los componentes de la sangre, siendo seleccionados los necesarios para su aplicación en medicina y devueltos al torrente sanguíneo el resto de componentes. La finalidad de la aféresis es la extracción de un componente sanguíneo destinado a la transfusión o para el tratamiento de algunas enfermedades que precisen la eliminación de un componente patológico de la sangre.

1.1.2 Componente Sanguíneo

Un constituyente terapéutico de la sangre (glóbulos rojos, leucocitos, plaquetas, plasma) que se pueda preparar mediante la centrifugación, filtración y congelamiento utilizando la metodología convencional de los bancos de sangre.

1.1.3 Donación de reposición

Cuando una persona pierde sangre en gran cantidad por un accidente o una operación, o tiene problemas de salud, puede que sea necesario que reciba una transfusión de sangre. Sin embargo, como la sangre humana es una sustancia que actualmente no se puede sintetizar, es necesario extraerla de otra persona, un donante de sangre.

1.1.4 La donación voluntaria de sangre

Se refiere a la práctica de donación de sangre proveniente de donadores exclusivamente voluntarios sin remuneración alguna de por medio. En contraparte se reconoce la donación de reposición en la que básicamente un familiar, amigo o conocido atiende una petición de donación de sangre en favor de alguna persona o paciente.

Es relevante distinguir a la donación voluntaria de sangre en la medida que representa una estrategia de donación que asegura un menor riesgo que la sangre captada sea portadora de alguna serología infecciosa o enfermedad transmisible vía transfusión.

En este sentido, la promoción y logró de la donación voluntaria de sangre no es un problema aislado de la seguridad sanguínea en muchos países del mundo y forma parte de la estrategia que los mismos pretenden seguir para alcanzar un mejor nivel en cuanto a su seguridad sanguínea.

La transmisión de enfermedades, vía transfusión de sangre, es aún un problema de salud pública que debe ser atendido por muchos países del orbe, cuya ocurrencia depende de una serie de factores diversos.

1.1.5 Almacenamiento

Sistema que permite guardar por un período que corresponde a la vida media de un producto sanguíneo en condiciones adecuadas a cada uno de ellos.

1.1.6 Donaciones Efectiva

Donaciones que generaron una bolsa de sangre apta para ser procesada.

1.1.7 Donaciones frustras

Personas que fueron seleccionadas como donantes, pero para quienes la extracción no se pudo realizar o se efectuó de manera incompleta ya sea por: dificultad de la vía venosa; reacción a la donación; la bolsa no se llenó completamente o se llenó demasiado; la bolsa no se agitó lo suficiente u otra causa y no será usada para preparar componentes.

1.1.8 Producto Sanguíneo

Cualquier producto terapéutico derivado de la sangre o plasma humana e incluye tanto a los componentes hábiles como a los derivados plasmáticos estables.

1.1.9 Pruebas pretransfusionales

Exámenes de inmunohematología que se deben realizar a los receptores de componentes sanguíneos que permitan asegurar una compatibilidad entre donantes y receptores. Las pruebas debe asegurar la compatibilidad de todos los anticuerpos con significación clínica.

1.1.10 Pruebas Inmunohematológicas

Exámenes de inmunohematología que se deben realizar a la sangre donada.

1.1.11 Solicitud de componentes sanguíneos al banco de sangre

Consiste en el proceso de solicitar al banco de sangre, desde cualquier área del hospital (Cuerpo de Guardia, Hospitalización, Quirófano, Consulta Externa), uno o varios componentes sanguíneos, para un paciente que requiera ser transfundido.

1.2 Descripción del flujo actual de los procesos.

Dentro de los procesos fundamentales que se llevan a cabo dentro de un banco de sangre están:

- ✓ Donación de sangre.
- ✓ Transfusión de sangre.
- ✓ Distribución y recepción de sangre

1.2.1 Donación de sangre.

El proceso se inicia cuando un donante se presenta al banco de sangre y solicita realizar una donación, seguidamente es atendido por una secretaria que le realiza una entrevista. En la cual registra sus principales datos personales y luego el donante es atendido por un médico que le realiza el examen físico, registrando también los resultados del mismo. Luego es enviado al laboratorio para realizarle los análisis de hemoglobina y las pruebas de serología. Todos estos datos, son registrados en la historia clínica del donante; siempre se le crea una nueva, para cada donación. Además se comprueba que el donante cumpla con el criterio de frecuencia mínima, que en el país es de tres meses. También, el médico revisa si en su anterior donación el donante tuvo algún contratiempo, un desmayo, fatiga, etc. Si todo resulta favorable, se acepta la donación y se archiva la historia clínica.

1.2.2 Transfusión de sangre

Existen transfusiones urgentes y planificadas. Las transfusiones urgentes se dan cuando ocurre cualquier incidente de improviso y hay que transfundir al paciente. Cuando esto ocurre solicita la sangre al banco de sangre, se le hacen las pruebas cruzadas para saber si son compatibles o no la sangre de la bolsa y si son compatibles inmediatamente es transfundido. Si no, la bolsa abierta se desecha y se toma una nueva bolsa a la que se somete a las mismas pruebas cruzadas que la anterior. Un día después se registra en el banco de sangre las órdenes de transfusiones donde se recogen los datos de la persona que será

transfundida, el motivo, la cantidad y otros datos de interés que se emitieron el día que se transfundieron los pacientes.

Las transfusiones programadas son en el caso de una operación. En este caso el paciente desde que se va a operar, se presenta al banco de sangre en el cual se le hace el Grupo sanguíneo y factor RH. Esto se registra en un libro para una posterior consulta. El día antes de la operación, se verifica con ayuda de los datos recogidos en este libro, que en el banco de sangre haya sangre disponible para que esa persona se pueda operar, de no ocurrir así se posterga la operación hasta que se disponga de la sangre necesaria.

Si se determina que la persona puede ser operada en el día indicado, se le informa que debe presentarse a operarse y luego de la operación si se determina que el paciente debe ser transfundido, este es sometido a las pruebas pretransfuncionales y luego transfundido, siendo registrada en el banco de sangre la orden de transfusión emitida el día que se puso la transfusión, al otro día de la operación.

1.2.3 Distribución y recepción de sangre

El banco de sangre es el área de salud que debe proporcionar la sangre que se necesite dentro del hospital, por ello debido al gran movimiento de unidades que puede ocurrir durante una jornada laboral, es necesario un estricto control de la distribución y recepción de sangre y la existencia en cada momento de unidades por grupo sanguíneo en la nevera de almacenamiento. Para ello se tendrán en cuenta las entradas y salidas de unidades del banco de sangre, pueden existir distintos tipos de entradas, como, donaciones aceptadas y unidades recibidas de otros centros, y distintos tipos de salidas como, unidades liberadas para transfusión, unidades enviadas a otros centros, y unidades vencidas o en fecha de vencimiento.

1.3 Descripción del sistema

Debido a los problemas generados a partir del manejo de la información en los bancos de Sangre que pueden traer consigo perdidas o mala gestión de las informaciones de los donantes y los datos y estadísticas de la sangre que aquí se almacena de los cuales la institución es responsable se hace necesario la implementación de un Sistema de Gestión de la Información para los Bancos de Sangre que automatice todos los procesos que se llevan a cabo como parte del trabajo cotidiano dentro de este departamento.

El sistema debe cumplir con un manejo óptimo y confiable de las informaciones de los pacientes y la sangre que entra y sale del banco y de los procesos dentro del mismo, una disminución de los recursos que se utilizan para el manejo de la información que se archiva, así como un considerable ahorro del

tiempo en que se realizan todas las actividades de este departamento. El sistema debe facilitar un mayor flujo de información, logrando que esta sea mucho más confiable y fácil de manipular. Además que exista mayor variedad y confiabilidad en los archivo con la información confidencial de los pacientes.

La aplicación se integraría al sistema de gestión hospitalaria que opera en el hospital, posibilitando una comunicación mucho más rápida y precisa entre las diferentes áreas del centro. Posibilitando la integración y comunicación con otros módulos logrando así, una mayor rapidez y agilidad al dar respuesta por parte del Banco de Sangre a un pedido realizado desde cualquier departamento. Al gestionar la disponibilidad de la sangre existente, la compatibilidad sanguínea de los pacientes entre muchas otras facilidades que serán ofrecidas con la implantación de este sistema.

Además, se dispondría de un mayor flujo de información de todas las acciones que se realizan en el Banco de Sangre, con todos los reportes disponibles. El sistema poseería, un mayor flujo de información mucho más organizada, de las donaciones hechas y de las transfusiones que son realizadas a cada paciente que se encuentra hospitalizado, así como de aquellos que son atendidos por urgencia.

1.4 Sistemas existentes vinculados al campo de acción

Se investigó sobre las principales aplicaciones Web y de escritorio, similares al que se desea desarrollar, caracterizándolas y describiendo sus principales funcionalidades. A continuación, se analizan sistemas desarrollados para el trabajo en Bancos de Sangre en distintas partes del mundo.

Un producto es el **Galen Banco de Sangre**, de la empresa cubana productora de software Softel, es un sistema automatizado integral para bancos de sangres.

Permite

- llevar un control del registro de donantes
- distribución de sangre y componentes sanguíneos
- producción del plasma a través de plasmaféresis
- producción de sueros,
- además cuenta con un módulo para el registro centralizado de donantes.

Características:

- Software propietario
- Desarrollado para una aplicación de Windows

- El lenguaje usado es Visual Basic 6.0, Delphi, lo que implica costos de desarrollo e instalación, en muchos casos elevados

DELPHYN, desarrollado en el Líbano.

Características:

- Es multiplataforma
- Software propietario
- Desarrollado para una aplicación Web
- Implementado en Java.

CARE2X software cuyo país productor es Alemania

Características

- Definido como software libre
- Multiplataforma
- Desarrollado sobre el lenguaje PHP
- Su producto es una aplicación Web.

Xblood es otro producto que a sido desarrollado para bancos de sangre. Fue desarrollado en México.

Características:

- Software propietario
- El producto es una aplicación de escritorio para Windows como el visto anteriormente.

Blood Bank Control System un producto desarrollado en los Estados Unidos para el trabajo en los Bancos de Sangre.

Características:

- Software propietario
- El producto es una aplicación de escritorio.

E-DELPHYN este es del país Emiratos Árabes Unidos para los bancos de sangre

Características

- Multiplataforma
- Software propietario
- Desarrollado para el trabajo sobre una aplicación Web.

TIMHEMO es una solución que ofrece TIMSA para Bancos de Sangre, es mexicana esta empresa y permite:

- Contar con la información en línea
- llevar un control minucioso de las unidades de sangre, receptores y donadores.
- facilitar y perfeccionar todos los procesos que se llevan acabo dentro de la Medicina Transfusional
- se automatizan los procesos lo que permite tener un mayor cuidado en la evaluación de donadores.
- Tener un control minucioso de las unidades de sangre, control sobre receptores y donadores y un rechazó automático en fin brindar seguridad y la mejor imagen en los resultados.

La solución cuenta con sonido audible para:

- Si el usuario llegara a introducir información fuera del rango preestablecido, con la finalidad de proveer el servicio más seguro para los pacientes.
- Detectar a través del inventario de unidades la disponibilidad de unidades NO VENCIDAS llevando así un control de la caducidad de unidades.

Versiones

CLIENTE - SERVIDOR

Consta de una red interna en la instalación donde la información se encuentra en línea; es decir todos los computadores que se encuentren dentro de la red están compartiendo la información según el nivel de acceso del usuario.

WEB

La versión Web es la más innovadora del mercado, provee el mismo servicio sin la necesidad de una inversión muy grande; lo único que necesita es una PC que tenga acceso a Internet y su clave. Es sencilla, fácil y segura.

La información se encuentra respaldada en más de 3 servidores (México, EUA, Europa) a la cual podrá acceder desde cualquier parte del mundo.

Con la investigación de estos sistemas se pudieron encontrar las tecnologías usadas para su implementación a si como las ventajas que les han traído las mismas para su puesta en marcha y para la obtención de los resultados.

Se tuvo en cuenta en el caso de los que fueron estudiados con mayor profundidad, las cosas que permiten, los módulos que desarrollan para en conjunto con el levantamiento de requisitos que se llevo a cabo en el hospital cubano tratar de cubrir todas las necesidades de los Bancos de Sangre, pretendiendo abarcar en el sistema la mayor cantidad de información posible para que el resultado de lo que se está construyendo sea el más óptimo.

Este estudio permitió llevar a cabo comparaciones que sirven para determinar que herramientas se van a usar dado los resultados que se han obtenido en estos software específicamente por la forma en que han sido implementados.

1.5 Tendencias y tecnologías actuales utilizadas.

Después de haber realizado un análisis de las necesidades encontradas en los bancos de sangre, y de las características del entorno donde se aplicará la solución propuesta; se realizó un estudio de las tendencias y tecnologías actuales posibles a emplear para adoptar la tecnología que aporte mayores ventajas en la elaboración de la solución. Estas tecnologías serán descritas en este epígrafe.

Se desarrollará es una **Aplicación Web,** que es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet o de una intranet y sus ventajas más significativas son:

Compatibilidad multiplataforma. Las aplicaciones Web tienen un camino mucho más sencillo para la compatibilidad multiplataforma que las aplicaciones de software de escritorio.

Actualización**.** Las aplicaciones basadas en Web están siempre actualizadas con el último lanzamiento sin requerir que el usuario tome acciones pro-activas, y sin necesitar llamar la atención del usuario o interferir con sus hábitos de trabajo con la esperanza de que va a iniciar nuevas descargas y procedimientos de instalación

Acceso Rápido. Las aplicaciones basadas en Web no necesitan ser descargadas, instaladas y configuradas. Se accede a desde una cuenta online y están listas para trabajar sin importar cuál es la configuración o hardware de la estación de trabajo.

Requiere de poca memoria**.** Las aplicaciones basadas en Web tienen muchas más razonables demandas de memoria RAM de parte del usuario final que los programas instalados localmente. Al residir y correr en los servidores del proveedor, esas aplicaciones basadas en Web, usan, en muchos casos, la memoria de las computadoras que ellos corren, dejando más espacio para correr múltiples aplicaciones al mismo tiempo sin incurrir en frustrantes deterioros en el rendimiento.

Varios usuarios concurrentes**.** Las aplicaciones basadas en Web pueden realmente ser utilizada por múltiples usuarios al mismo tiempo. No hay necesidad de compartir pantallas o enviar instantáneas cuando múltiples usuarios pueden ver e incluso editar el mismo documento de manera conjunta.

Mono es una plataforma de desarrollo de código abierto basada en el .NET Framework. Permite ejecutar y desarrollar aplicaciones modernas basadas en los estándares ECMA/ISO. Mono puede ejecutar aplicaciones hechas para los framework .NET y Java. (*Informacion de Mono*)
Mono permite a los desarrolladores construir aplicaciones Linux y multiplataforma de alta productividad. La implementación de Mono cumple los estándares ECMA para C# y la infraestructura de lenguaje común (Common Language Infraestructure, CLI).

Acoge varias licencias libres (X11, LGPL y GPL) y tiene una comunidad de desarrolladores activa, está esponsorizado por Novell y es la base para muchas aplicaciones. Incluye compiladores, un intérprete compatible con el CLR de ECMA y un conjunto de librerías. Las librerías cubren la compatibilidad con Microsoft .NET (incluyendo ADO.NET, System.Windows.Forms y ASP.NET). Mono posee librerías adicionales y otras de terceras partes.

Esto significa por ejemplo, que si defines una clase que haga una manipulación algebraica en C#, esa clase puede ser reutilizada en cualquier lenguaje que soporte el "CLI". Puede crear una clase en C#, una subclase en C++ e instanciar esa clase en un programa en Eiffel.

La herramientas de desarrollo desde Windows para desarrollar una aplicación es utilizando el Visual Studio como en este caso. En el lado de Linux existe el MonoDevelop, el cual es un IDE basado en el SharpDevelop. Con el objetivo de lograr un proyecto totalmente libre y multiplataforma luego de desarrollarse la aplicación sobre Visual Studio se correrá sobre Mono.

También se hará uso de las **hojas de estilo en cascada** (*Cascading Style Sheets*, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirá de estándar para los agentes de usuario o navegadores.

Se utilizará **ASP.NET** que no es más que un conjunto de tecnologías de desarrollo de aplicaciones Web comercializado por Microsoft. Es usado por programadores para construir sitios Web domésticos, aplicaciones Web y servicios XML. Forma parte de la plataforma .NET de Microsoft y es la tecnología sucesora de la tecnología Active Server Pages (ASP).

Para la comunicación de este con otros módulos y sistemas que se usen dentro de los hospitales se hacen uso de **servicios Web** (en inglés *Web service*) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferente y ejecutada sobre cualquier plataforma pueden utilizar los servicios Web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

El proveedor que se utilizará para el acceso a datos es Npgsql proveedor ADO.NET escrito completamente en C# que permite trabajar desde cualquier aplicación .NET (usando VB.NET, C#, C++, etc.) con Postgresql 7.x y 8.x que es le gestor de base de datos que se utilizará en el sistema.

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.
Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes (más notablemente de Delphi y Java). C# fue diseñado para combinar el control a bajo nivel de lenguajes como C y la velocidad de programación de lenguajes como Visual Basic.

Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado sobre ella, por lo que programar usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes, disponibles en .NET, ya que C# carece de elementos heredados innecesarios.

Las principales características que hacen del lenguaje una elección óptima: son:

- Sencillez, independencia de tipos: El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad de código.
- Modernidad: C# incorpora en el propio lenguaje elementos que se han mostrado útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++.
- orientación a objetos: Como todo lenguaje de programación, C# es un lenguaje orientado a objetos, no admite ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.
- orientación a componentes: La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).
- seguridad de tipos, no se pueden usar variables no inicializadas, versionable: C# incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijo previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.
- eficiente, compatible: Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente código escrito en C# fragmento de código escritos es estos lenguajes, sino que el CLR también ofrece, a través de los llamados Plataform Invocation Services (Pinvoke).

Aunque C# forma parte de la plataforma .NET, ésta es una interfaz de programación de aplicaciones; mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma existen los siguientes compiladores e IDEs para el lenguaje C#:

- Microsoft .NET *framework* SDK incluye un compilador de C#, pero no un IDE.
- Microsoft Visual C#, IDE por excelencia de este lenguaje, versión 2002, 2003 y 2005.
- SharpDevelop, es un IDE libre para C# bajo licencia LGPL, muy similar a Microsoft Visual C#.
- Mono, es una implementación GPL de todo el entorno .NET desarrollado por Novell. Como parte de esta implementación se incluye un compilador de C#.
- Delphi 2006, de Borland Software Corporation.
- DotGNU Portable.NET, de la Free Software Foundation.(*Información de CSharp*)

AJAX. es la unión de varias tecnologías que juntas pueden lograr cosas realmente impresionantes, incorporando: presentación basada en estándares usando XHTML y CSS; exhibición e interacción dinámicas usando el Document Object Model; intercambio y manipulación de datos usando XML and XSLT; recuperación de datos asincrónica usando XMLHttpRequest; y Javascript para unir todo el contenido.

AJAX, en resumen, es el acrónimo para Asynchronous Javascript + XML y el concepto es: Cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en *background*, los datos que son usados para actualizar la página solo re-renderizando la página y mostrando u ocultando porciones de la misma.
Ofrece 10 ventajas que se pueden denominar razones para usar estas tecnologías y que provocan que AJAX sea la palabra de moda en la comunidad desarrolladora de aplicaciones WEB.

Basado en los estándares abiertos

Ajax está formado por las tecnologías Javascript, HTML, XML, CSS, y XML HTTP Request Object, siendo este último el único que "no es" estándar pero es soportado por los navegadores más utilizados de internet como son los basados en Mozilla, Internet Explorer, safari y opera.

Usabilidad

Permite a las páginas hacer una pequeña petición de datos al servidor y recibirla sin necesidad de cargarla página entera. El incremento de las actualizaciones "on the fly" elimina el tener que refrescar el navegador, algo bastante apreciado a la hora de operar en una aplicación web.

Válido en cualquier plataforma y navegador

Internet Explorer, los basados en Mozilla y Firefox son los que se llevan la palma en el mercado de internet y además son los navegadores en los que es más fácil programar aplicaciones Web AJAX, pero ahora es posible construir aplicaciones web basadas en AJAX para que funcionen en los navegadores más modernos. Es una de las razonas más importantes por las que AJAX se ha vuelto tan popular. Aunque si bien muchos desarrolladores sabían que era posible usarse años atrás con Internet Explorer, no era viable realizarse. Ahora ya es posible su avance gracias a Mozilla y Firefox.

Beneficia las aplicaciones web

AJAX es la cara del presente en las aplicaciones web - las aplicaciones web conllevan ciertos beneficios sobre las aplicaciones sobre escritorio (aplicaciones que dependan de un sistema operativo, librerías, lo que se entiende por programas compilados). Esto incluyo un menor coste de creación, facilidad de soporte y mantenimiento, menores tiempos a la hora de desarrollarlas, y sin necesidad de instalaciones; éstas son algunas de los beneficios que han llevado a las empresas y usuarios el adoptar aplicaciones web desde mediados de los 90. AJAX solo ayudará a las aplicaciones web a mejorar y conseguir un mejor resultado de cara al usuario final.

No es difícil su utilización

Porque AJAX está basada en los estándares que han sido utilizados durante muchos años, muchos desarrolladores web han tenido que utilizar las tecnologías que las aplicaciones requieren. Esto significa que no es un gran esfuerzo el aprendizaje de los desarrolladores el pasar de un simple código HTML y aplicaciones web a una potente aplicación AJAX. También significa que los desarrolladores pueden actualizar poco a poco las interfaces de usuario hacia unas interfaces con AJAX; no necesita una re-escritura de la aplicación entera, se puede hacer incrementalmente.

Compatible con Flash

Muchos desarrolladores tienen serias dudas sobre usar Flash o AJAX. Definitivamente hay ventajas y desventajas en ambas tecnologías según la situación que se de pero también hay muchas posibilidades y muy buenas para que amabas funcionen en conjunto.

Adoptado por los "gordos" de la tecnología web

La difusión de AJAX en los líderes de la industria de internet prueba que el mercado acepta y valida el uso de esta tecnología. Todo el mundo está migrando hacia AJAX incluyendo Google, Yahoo, Amazon, Microsoft (por nombrar unas pocos). Google Maps fue lo que captó la atención de los desarrolladores web.

Cuando empezaron a investigar como google era capaz de llevar esa increíble herramienta dentro de un navegador sin necesidad de ningún tipo de plug-in, encontraron que AJAX estaba detrás del tema.

Web 2.0

El movimiento Web 2.0 está cada vez más en auge y dando quebraderos de cabeza de muchos programadores, usuarios, y vendedores. Esto está ayudando la adopción de AJAX. Las interfaces de AJAX son un componente clave de muchas de las aplicaciones Web 2.0, como puede ser Back Pack (un organizador de disco online en entorno Web) y Google Maps. Afortunadamente gracias bombo que se le está dando, acelerará la adopción de AJAX y la los beneficios de su uso lo mantendrá en escena. Una de las claves principales de Web 2.0 es el usar la red como plataforma para el desarrollo de aplicaciones, en vez de simples páginas web. Siendo importante la iteración de los usuarios con la aplicación en sí.

Es independiente del tipo de tecnología de servidor que se utilice

Así como AJAX funciona en cualquier navegador, es perfectamente compatible con cualquier tipo de servidor estándar y lenguaje de programación Web. PHP, ASP. ASP.Net, Perl, JSP, Cold Fusión. El ser completamente compatible el desarrollo en éstas tecnologías ha ayudado a AJAX a que vaya cada vez más en auge.

Mejora la estética de la web

Con AJAX se puede interactuar la imaginación del desarrollador con la usabilidad de una aplicación web de forma que se pueda realizar una aplicación que si no estuviera dentro de un navegador, podría pasar por una aplicación normal de escritorio. (*Ajax*)

1.6 Herramientas de desarrollo.

1.6.1 Visual Studio 2005

Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones Web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C++, Visual C# y Visual J# utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes aprovechan las funciones de .NET Framework, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y Servicios Web XML.

Incorpora nuevas y mejoradas funciones de productividad: Configuración del IDE, importar y exportar configuraciones, listas de tareas, lista de errores, teclas de método abreviado Brief y Emacs.

Visual Studio presenta un nuevo diseñador de páginas Web que incluye muchas mejoras para la creación y edición de páginas Web de ASP.NET y páginas HTML. Proporciona una forma más fácil y rápida de crear páginas de formularios Web Forms que en Visual Studio .NET 2003.

La vista Diseño del diseñador HTML incluye muchas mejoras que admiten las nuevas funciones de ASP.NET o facilitan el diseño WYSIWYG de páginas Web. La edición basada en tareas mediante etiquetas inteligentes le guía durante la ejecución de los procedimientos más comunes con controles, como el enlace de datos y la asignación de formato.

Puede editar visualmente las nuevas páginas principales de ASP.NET. La edición de plantillas se ha mejorado para facilitar el trabajo con controles de datos, así como con nuevos controles como el control Login. Editar tablas HTML para el diseño o mostrar la información en columnas ahora es más fácil e intuitivo.

1.6.2 SharpDevelop

SharpDevelop es un Entorno integrado de desarrollo libre para los lenguajes de programación C#, Visual Basic .NET. Es usado típicamente por aquellos programadores de los citados lenguajes, que no desean o no pueden usar el entorno de desarrollo de Microsoft, el Microsoft Visual Studio.

Hay disponible un port para Mono/Gtk#, llamado MonoDevelop, el cual funciona en otros sistemas operativos. Para el completado automático de código, la aplicación incorpora sus propios parsers. La versión 1.1 de la aplicación puede importar proyectos de Visual Studio .NET. La versión 2.1 ya es capaz de editarlos directamente.
Es un proyecto que comenzó desde los comienzos del lenguaje C# en el año 2000, y cada ves se pone mejor, el mes pasado salió su ultima versión, algunas de sus características, soporta C#, Visual Basic .NET, ASP.NET, XML. Es de diseño visual al igual que el Visual Studio, y bastante similar, es compatible con Net Framework 1.1, 2.0, Compact Framework y Mono (el mismo de Linux), viene con debugger, configuración de proyectos, además de ser escrito enteramente en C#.

SharpDevelop Incorpora un diseñador de Windows forms, Completado de código. Soporta el uso de la combinación de teclas Ctrl + Espacio, depurador incorporado, herramientas para "Ir a Definición", "Encontrar referencias" y "renombrado", escritura de código C#, ASP.NET, ADO.NET, XML y HTML, Llaves inteligentes en la escritura de código.

En este capítulo, se han analizado los sistemas que se encuentran desarrollados para dar solución a la problemática existente en los Bancos de Sangre, a nivel mundial, aunque estos no cumplen con las expectativas que hoy se plantean en el país. Por lo que, luego del estudio de los mismos fue posible percatarse que se necesita desarrollar un software con mayor calidad.

Para lograr el objetivo se realizó el estudio de las tendencias y tecnologías que se utilizaran, demostrando cuántas ventajas ofrecen, por lo que se considera que la elaboración de este capítulo ha sido muy importante para el desarrollo de la tesis.

Además, servirá como base para fundamentar que lo que se plantea, es más que la mejora de la problemática existente en los Bancos de Sangre; es el desarrollo de un software que brinde las facilidades de las tecnologías que han sido seleccionadas y que supere las desventajas de los software analizados en este capítulo y cumpla con todos los requerimientos funcionales y no funcionales que exige el sistema.

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

En este capítulo se hizo una breve pero profunda valoración del diseño propuesto por el analista del sistema exponiendo las principales ventajas que ofreció la obtención del mismo. [Epígrafe 2.1]

Fueron analizados además, los componentes o módulos ya existentes y que pudieran ser usados, lo que resulta muy ventajoso dado que la reutilización ofrece ventajas como la reducción de los esfuerzos de desarrollo y el mantenimiento y mejora de la seguridad, la eficiencia y la consistencia de los diseños. [Epígrafe 2.2]

Se realiza también la descripción de un algoritmo cuya funcionalidad es una de las más importantes en el trabajo en los Bancos de Sangre y que dada su complejidad requirió de un profundo análisis y de un trabajo de desarrollo bastante difícil. Por este motivo, se dedica parte de este capítulo a que usted pueda lograr un mayor entendimiento mediante la explicación. [Epígrafe 2.4]

En programación, una estructura de datos es una forma de organizar un conjunto de datos elementales (un dato elemental es la mínima información que se tiene en el sistema) con el objetivo de facilitar la manipulación de los mismos como un todo o individualmente define la organización e interrelación de estos y un conjunto de operaciones que se pueden realizar sobre el. Es por esto que su uso se impone en la implementación del sistema y se dedicó el epígrafe 2.5 de este capítulo para su explicación.

2.1 Valoración crítica del diseño propuesto por el analista.

La obtención del diseño propuesto por el analista, resultó muy importante, pues permitió adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.

También posibilitó crear una entrada apropiada y un punto de partida para las actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.

Se obtuvieron además, diagramas como los de clases de diseño que mostraron la parte estática del sistema, la representación de las clases y sus relaciones cosa que facilita en gran medida la implementación de las mismas dada sus definiciones.

También los diagramas de interacción que muestran las interacciones mediante transferencia de mensajes entre objetos o subsistemas.

En el diseño se aplicaron patrones de diseño como es el caso del Patrón Fabricación Pura que proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas.

Motivación

- Un sistema debe ser independiente de cómo se crean, componen y representan sus productos
  - Un sistema debe ser configurado con una familia de productos de entre varias
  - Una familia de objetos relacionados está diseñada para ser usada conjuntamente, y es necesario hacer cumplir esta restricción
  - Quiere proporcionar una biblioteca de clases, y sólo quiere revelar sus interfaces, no sus implementaciones

Ventajas

- Aísla las clases concretas para el cliente
- Facilita el intercambio entre familias de clase
- Promueve la consistencia

Desventajas

- No es sencillo agregar elementos a la factoría.
- Requiere modificar la interface
- Requiere modificar las subclases

Utilizando este patrón, se gana soporte a una Alta Cohesión pues se logra distribuir las responsabilidades en clases de granularidad fina, centradas en un conjunto muy específico de tareas afines.

Estos patrones contribuyen a reutilizar diseño, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. Mejoran la flexibilidad, modularidad y extensibilidad, factores internos e íntimamente relacionados con la calidad percibida por el usuario.

2.2 Componentes o módulos ya existentes y que puedan ser reusados.

Como se explicaba en la introducción la reutilización resulta sumamente importante pues brinda ventajas como el ser clave a la hora de reducir los tiempos y costes de desarrollo, parámetros que marcan la diferencia entre una arquitectura competitiva y otra que no lo es.(*Reutilización*)

Un módulo que puede ser reusado es el framework Hermes de HL7.

Health Level Seven (HL7)

HL7 es una organización internacional, iniciada en los Estados Unidos en 1987, que pretende promover el desarrollo y evolución del estándar para el formato de datos e intercambio de información entre diferentes sistemas de información de salud.

Está abierta a todos los diferentes actores del ámbito de las tecnologías de la información y la salud (prestadores de servicios de salud, desarrolladores de software, consultores, usuarios finales, organizaciones gubernamentales, universidades y otras organizaciones) y desarrolla estándares por consenso en un entorno abierto.

En la actualidad, los sistemas informáticos de los servicios de salud de las comunidades, promocionan el uso del estándar *XML/HL7***.** Esta tecnología, garantiza la transferencia de información clínica entre comunidades autónomas y entre hospitales, desde los resultados de una prueba diagnóstica hasta el historial completo de un paciente. Ejemplo de esto: Software AG promueve el uso de XML/HL7, un estándar basado en XML específico para el sector sanitario, que hace posible la interoperabilidad entre los sistemas informáticos de distintos servicios de salud. De esta forma, cualquier médico o ciudadano tiene la posibilidad de acceder a su historia clínica única si necesita de atención especializada o requiere de una intervención quirúrgica fuera de su comunidad.

Bajo este protocolo de intercambio de información estándar se facilita la interconexión entre los distintos sistemas informáticos de las comunidades autónomas para garantizar así un flujo rápido y seguro de la información clínica del paciente.

Además se hará uso de Chameleon que próximamente será adquirido por la universidad.

Chameleon

Los componentes de mensajería *Active-X* para cibersalud incorporan en objetos los mensajes de la versión 2.3 de HL7 y facilitan la conexión de varios sistemas. Estas aplicaciones sólo tienen que solicitar el objeto idóneo (por ejemplo una orden de admisión o de medicación), agregar los elementos de datos apropiados y enviar de forma transparente el objeto de que se trate a las aplicaciones configuradas para

recibirlo. Una arquitectura flexible posibilita la migración entre los motores de interfaz y los sistemas tradicionales a aplicaciones basadas en componentes y conectadas en red.

Hoy día, pueden adquirirse productos HL7. *HL7 Chameleon* es uno de los muy conocidos productos. Su objetivo principal es que separa la interfaz de mensaje de las diferentes implementaciones de HL7, lo que permite que una aplicación escrita con una interfaz soporte un gran número de diferentes versiones de HL7. Permite afrontar los problemas que puede plantear el aislamiento del código de aplicación respecto a la estructura de datos HL7 sin codificar. Chameleon utiliza el archivo de definición de mensaje (VMD) como interfaz que actúa como puente entre el sistema y el mundo HL7.

Soporta las versiones HL7 2.1 a 2.6 y la versión 3 cuando se encuentre disponible.
Es importante aclarar que dicha interfaz se crea dentro del VMD usando tablas, las cuales se usan como objetos con los cuales se interactúa. Los datos son mapeados desde los mensajes entrantes y viceversa.
Dicha herramienta se encuentra disponible para las plataformas Windows 95/98/2000/XP, Windows Server 2003, Windows NT 3.51 y superior, Mac OS X, Solaris, GNU-Linux, SCO Unix, AS400, Tandem o cualquier sistema operativo que soporte un compilador C++ que cumpla con la norma ANSI 92.
Está equipada con un grupo de aplicaciones de gran utilidad que de ser usadas equilibradamente reducen en gran medida el esfuerzo empleado en la creación y mantenimiento de las interfaces del sistema. Las aplicaciones mencionadas con anterioridad son HL7 Listener, HL7 Simulator, Message Browser y herramientas accesibles desde la línea de comandos (msgtransform, msgxml y msgdiff).

Presenta soporte para los lenguajes más utilizados en la actualidad: Java, C++, C#, Visual Basic, Delphi, .NET más soporte ActiveX para lenguajes asociados a la plataforma Windows como Power Builder, FoxPro y Microsoft Access.

Chameleon ha estado en el mercado por más de ocho años y se encuentra desplegado en más de 5000 sitios, además cuenta con una lista de clientes de prestigio internacional que avalan su reputación: Patient Keeper, IBM , AccuImage, Veteran Affairs, University Health Network (UHN), Blue Cross Blue Shield Association, All Meds, + NUTH, LockheedMartin, Aramark, General Electric Medical Systems y SECTRA.

Middleware: Es una subcapa con la función de acceder al repositorio de datos, es encargada de ejecutar la lógica de acceso a datos, contiene dos paquetes de clases:

Repositorios: Los repositorios son los encargados de manejar las colecciones de Objetos Comunes (Entidades de la BD representados como objetos) y realizar operaciones sobre ellas. La imagen que a continuación se muestra enseña las responsabilidades de un repositorio.

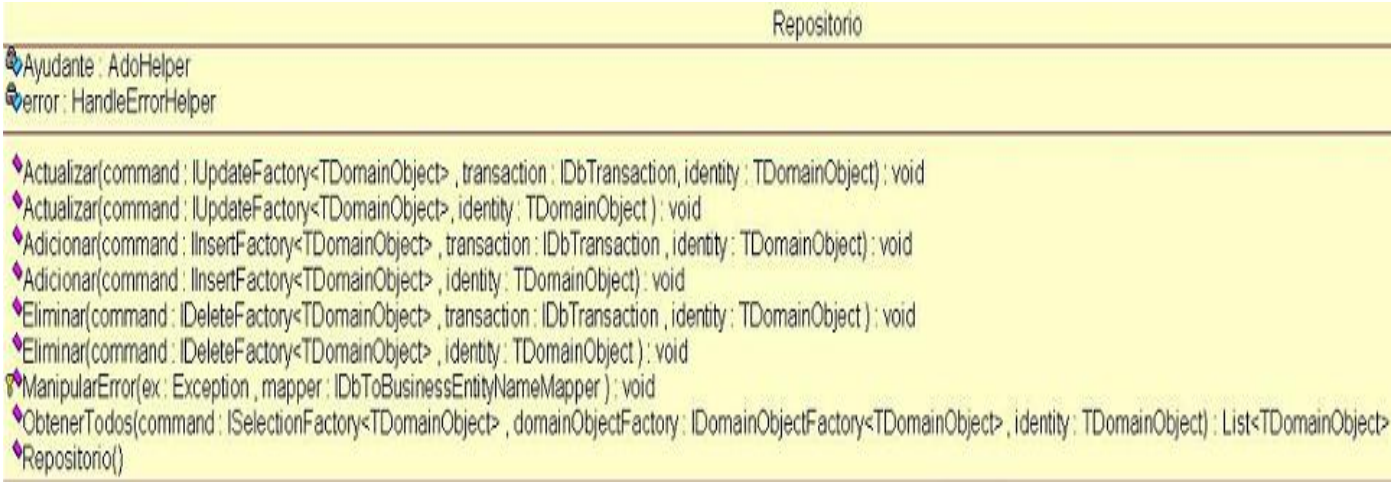

Fig.1 Repositorio

Fábricas de Objetos: Paquetes de clases que contiene la funcionalidad necesaria para acceder a la base de datos y realizar las operaciones que se explican a continuación. Por cada entidad mapeada desde la base de datos, existen cuatro clases que heredan de las interfaces ISelectionFactory: encargada de llevar a cabo el proceso de selección de una entidad determinada en la base de datos, IInsertFactory: encargada del proceso de inserción, IDeleteFactory: Encargada de suprimir una entidad determinada, IUpdateFactory: encargada de actualizar atributos de los objetos en la BD, existe además por cada entidad otra clase que hereda de la clase interfaz IDomainObjectFactory y es la encargada de realizar el proceso de mapeo de las entidades (convertir tupla a tupla el resultado de un proceso de selección en la entidad a la que corresponde).

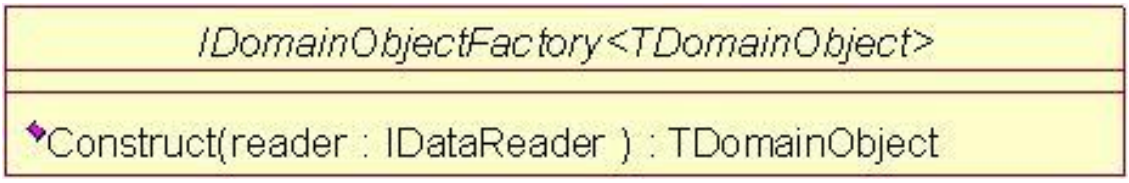

Fig.2 Fábricas de Objeto

Dentro de los módulos que se desarrollan en el proyecto productivo GeHos. Existen dos de los cuales se vuelve necesaria la reutilización de fragmentos de código.

El módulo **Inscripción Admisión** es uno, pues es el encargado se recopilar todos los datos de los pacientes que llegan al hospital cosa que no es más que crear las historias clínicas (HC) de los pacientes cuando arriben al mismo, además de gestionar todo lo que tiene que ver con ingresos, egresos y los movimientos que se realizan en el hospital en caso de ser necesario.

Por lo que una vez hechas las conexiones de manera directa a través de la implementación de una clase por parte de ese módulo que permita eso, se obtendrá toda la información que ha sido recolectada cuando se crea la historia clínica, datos que se podrán reusar, mediante la integración que se llevara a cabo en la capa de negocio de los módulos del sistema.

El otro módulo es **Configuración**, posee tablas donde se recopila información referente a los médicos, enfermeros y funcionarios, los datos generales de los usuarios del sistema, a los diagnósticos, entre otros. Dominan una lista donde se representan cualidades, estados, tipos de clasificaciones y características específicas que puede presentarse a un determinado paciente.

2.2.1 Estrategias de integración.

Todo el código dentro un mismo componente utiliza llamadas a métodos o eventos de forma directa. La comunicación entre diferentes componentes se realiza mediante llamadas a servicios web o de forma directa a nivel de negocio, en caso de utilizarse servicios web, la información que es transmitida debe cumplir con los estándares internacionales que hay establecidos para facilitar la integración entre nuevos componentes y otros sistemas hospitalarios.

La base de datos es accedida de forma directa mediante controladoras y los componentes reusados son integrados mediantes interfaces sencillas.

2.3 Estándares de codificación

Cuando se trabaja en equipo es necesario hacer código legible y entendible no sólo para quien lo escribe, sino también para quien lo lee, y para eso es necesario tener en cuenta varios aspectos:

- ✓ Las cláusulas, es decir, la notación que se utilizará para nombrar cada uno de los identificadores que se declaran
- ✓ La estructura del código en sí, es decir, lo referente a las tabulaciones y los espacios entre líneas, y dentro de las líneas, los espacios entre los operadores y estructuras que componen el lenguaje en que se programó.

2.3.1. Las cláusulas generales

Para nombrar identificadores es recomendable:

- Hacerlo en un solo idioma, en este caso: ***el español***.
- No usar números, ñ o caracteres especiales como @, $, o _ (underscore).
- Usar la notación Pascal o Camello para cada uno de los tipos de datos que se declaren.

2.3.1.1 Notación Camello

Se usa para denotar variables y parámetros. En esta notación, si el identificador es una palabra simple se escribe todo con minúscula, pero si es compuesta, la primera letra de todas las palabras que viene a continuación de la primera comienza con mayúscula. Se usa para denotar ***variables y parámetros***, y en ambos casos la primera palabra debe ser un sustantivo que describa claramente al identificador y las otras palabras a continuación deberán ser adjetivos.

Ejemplo:

```
Prívate Paciente paciente;
        Ó
Prívate Paciente pacienteAdmitido;
```

2.3.1.2- Notación Pascal

En la notación Pascal, si el identificador es simple, el primer carácter se escribe con mayúscula y el resto con minúscula; si el identificador es una palabra compuesta, la segunda palabra debe empezar con mayúscula también. Pero con esto no basta, también es necesario seguir algunas convenciones específicas para los distintos tipos de datos que se mencionan a continuación:

Espacios de nombres (namespaces)

No deben contener espacios y deben definir claramente el conjunto que representan.

Ejemplo:

```
Namespace ReglasDelNegocio
{
    //…
}
```

Clases

Los nombres de las clases deben ajustarse a la entidad que representan, y su primera palabra debe ser un sustantivo. Si con una sola palabra no se puede nombrar dicha entidad, la segunda palabra debe ser un adjetivo, a menos que la palabra sea compuesta.

Ejemplo:

```
Public class Jugador
{
    //…
}

Public class JugadorProfesional
{
     //….
}
```

Constantes

Para denotar las constantes se siguen las mismas cláusulas que las clases.

Interfaces

Para denotar interfaces, se siguen las mismas reglas que las clases, pero el primer carácter del identificador debe ser una 'I' (i mayúscula).

Ejemplo:

```
Public IMedico
{
     //…
}
```

Métodos

Los nombres de métodos deben describir la acción que realizan, y si el identificador es compuesto, la primera palabra debe ser el infinitivo de la acción.

Ejemplo:

```
Public void CambiarEstado (string nuevoEstado)
{
```

```
        //…
    }
```

Propiedades (Properties)

Si modifican o devuelven algún atributo perteneciente a una clase, debe tener el mismo nombre del atributo, pero su primera letra debe ser mayúscula. De otra forma deben seguir las cláusulas de los métodos.

Ejemplo:

```
Public string Nombre
{
   //…
}
```

Componentes

Para denotar los componentes se debe mantener la primera palabra que predefine Visual Studio para ellos y agregarle una palabra que empiece con mayúscula, que defina la acción que realiza o los datos que representa.

Ejemplo:

```
TextBoxNombre
ButtonAceptar
```

2.3.2 Estructura del código

Dentro de cada espacio de nombre, clase, interface, propiedad, método o evento, el código debe cumplir con las siguientes condiciones:

- Una línea para el identificador.
- Una línea para abrir llave.
- Una línea para cerrar llave.
- El margen entre las llaves y las sentencias debe ser de un tab.

Ejemplo:

```
Public string Indicaciones
{
        Get
        {
```

```
                Return this.indicaciones;
            }
            Set
            {
                this.indicaciones = value;
            }

        }
```

Dentro de cada condicional o estructura de control el código deberá cumplir con las siguientes condiciones:

- Una línea para abrir llave.
- Una línea para cerrar llave.
- El margen entre las llaves y las sentencias debe ser de un tab.

Ejemplo:

```
For (int i=0; i<edades.Length; i++)
{
    If (edades[i] == 5)
    {
        //….
    }
}
```

2.4 Descripción de los algoritmos no triviales a implementar.

Se denominó algoritmo no trivial a los algoritmos cuyo análisis e implementación hayan resultado complejos, a los que se le haya dedicado un poco más de esfuerzo, por lo que requieren ser descritos en un epígrafe de este capítulo para explicar como fue que se pensó desde un primer momento y todo lo que fue necesario investigar y desarrollar para que este método cumpla con las funcionalidades que requiere.

Como *algoritmo no trivial* a usar en la implementación de la tesis se encontró: El **Algoritmo de asignación de unidades por solicitud.**

Antes de comenzar a explicar las características del algoritmo se realizará una breve explicación de conceptos fundamentales en los que fue necesario basarse para la implementación del mismo.

La solicitud de componentes sanguíneos al banco de sangre consiste en el proceso de solicitar al banco de sangre, desde cualquier área del hospital (Cuerpo de Guardia, Hospitalización, Quirófano, Consulta Externa), uno o varios componentes sanguíneos, para un paciente que requiera ser transfundido.

Los tipos de solicitud que existen: La solicitud electiva o programada que es la que se hacer por parte del Bloque quirúrgico una vez que se ha programado una operación a un paciente a corto o largo plazo, y que puede ser aplazada por la disponibilidad existente en el Banco de sangre del tipo que se pide.
La solitud urgente que es la que se realiza por cualquier departamento del hospital que la requiera ante el arribo al hospital de un caso de urgencia donde se requiera sangre o algunos componentes de esta para uno o varios pacientes.

Los limites inferiores o Stock es el número total de unidades de componentes almacenados en condiciones adecuadas a cada uno de ellos para responder a la demanda de un centro.

Para realizar transfusiones, deben tomarse medidas para asegurar la compatibilidad de los grupos sanguíneos del donante y el receptor, para evitar reacciones hemolíticas posiblemente fatales.
Los donantes de sangre y los receptores deben tener grupos compatibles. El grupo O- es compatible con todos, por lo que quien tiene dicho grupo se dice que es un *donante universal*. Por otro lado, una persona cuyo grupo sea AB+ podrá recibir sangre de cualquier grupo, y se dice que es un *receptor universal*. La tabla que sigue indica las compatibilidades entre grupos sanguíneos. [Fig. 1] Por ejemplo, una persona de grupo A- podrá recibir sangre O- o A- y podrá donar a AB+, AB-, A+ o A-.

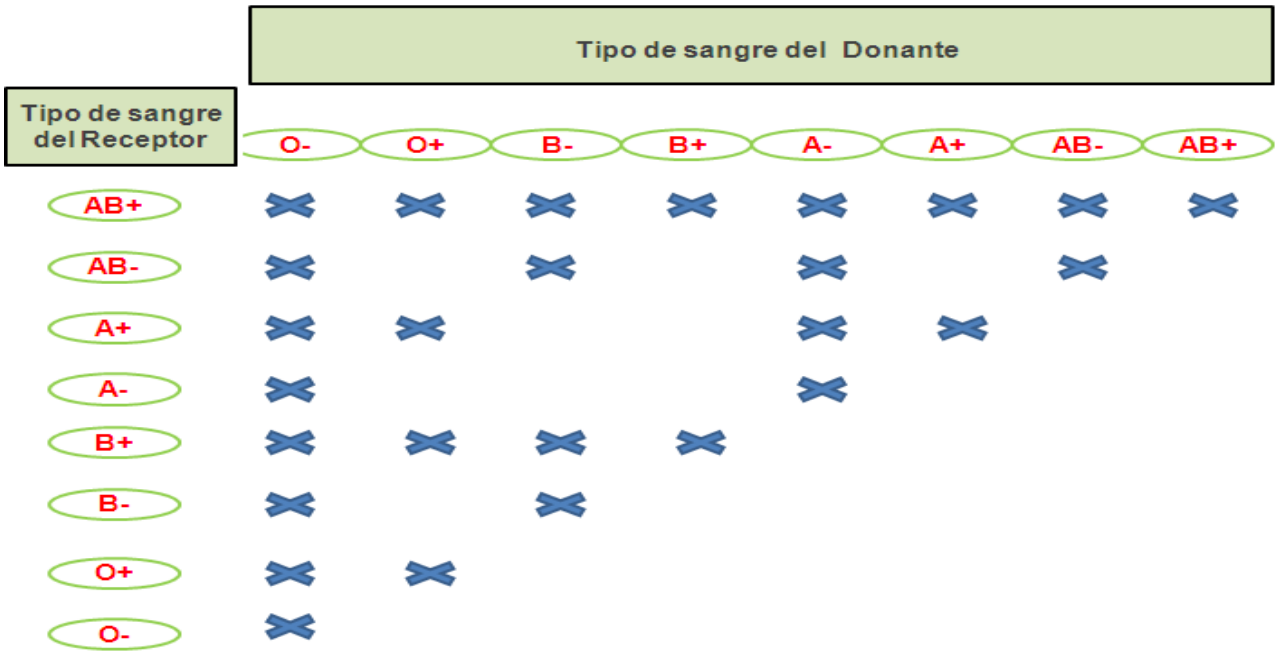

Fig. 3: Tabla de compatibilidades de tipos de sangre

La distribución de los grupos sanguíneos en la población humana no es uniforme. El más común es O+, mientras que el más infrecuente es AB-. Además, hay variaciones en la distribución en las distintas sub_poblaciones humanas. Como muestra la Fig. 4.

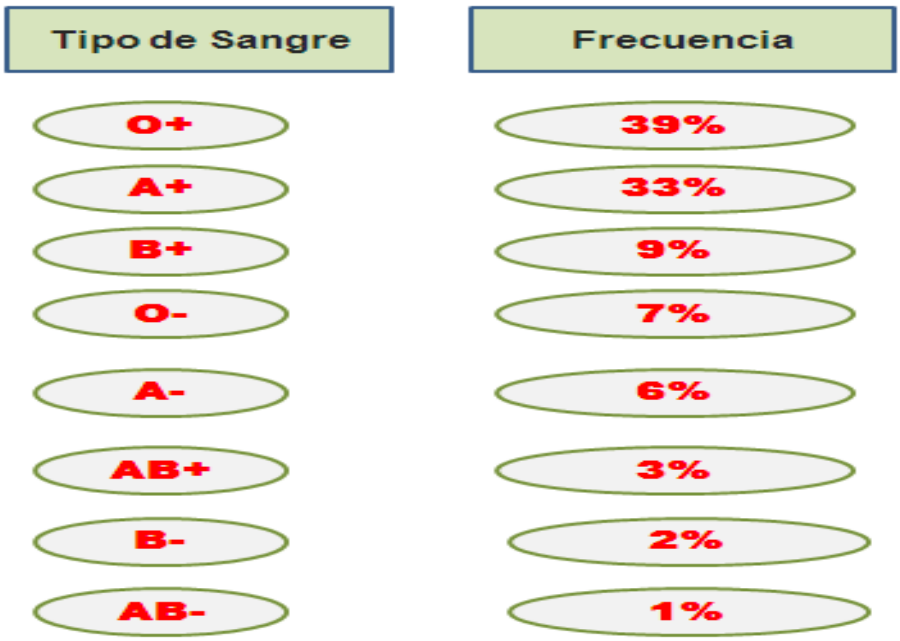

Fig.4 Distribución de frecuencia de los tipos de sangre

La fecha de caducidad de las bolsas de sangre es otro de los factores importantes que influyen en la realización de este método pues estas cuentan con periodo corto de disponibilidad, se encuentran en un rango de 22 a 35 días según el método de extracción que haya sido usado para obtener la sangre.

La reintegración o devolución de unidades por parte de los que realizaron el pedido de la misma es otra de las cosas a tener en cuenta dado que una vez salidos del sistema la bolsa de sangre se le cambia el estado a la misma por lo que ante la devolución de la bolsa hay que entrarla al sistema como una bolsa disponible y tener en cuenta su fecha de caducidad para próximos pedidos.

La complejidad del algoritmo está dada por:

- La necesidad de hacer una asignación en tiempo y del todo óptima para el receptor de la sangre, que previamente ha sido donada.
- La cantidad de prioridades que se deben tener en cuenta a la hora de hacer entrega de una bolsa de sangre, que resulte del todo compatible para el paciente que será transfundido, que no haya caducado y que no sobrepase los límites inferiores de stock que han sido establecidos en el banco de sangre.

Para la realización del algoritmo resultó necesaria la creación de dos procedimientos almacenados, el primero, se refiere a que pasando una fecha devuelva una lista de solicitudes que han sido hechas para esta fecha, y el otro a que es necesario entrarle como dato, el o los componentes que se necesitan, la cantidad de los mismos, el grupo sanguíneo y el factor RH. Una vez entrados estos datos, se creará una lista de las bolsas que se encuentren disponibles con estas características ordenadas por la fecha de vencimiento de las mismas tratando de lograr que las bolsas que se encuentren más cerca de vencerse que se sean compatibles con la sangre del receptor sean utilizadas.

A través de la figuras 5, 6, 7, y la fig. 8 se puede observar como ha sido implementada la solución al algoritmo que se trata: ¨Asignación por solicitud¨ en esta primera iteración se tendrá solución:

- Para las solicitudes urgentes, si hay disponibilidad de sangre en el Banco del mismo grupo sanguíneo y factorRH que se ha pedido y en caso de que no haya se enviara del tipo de sangre universal, O- que es compatible con todos los grupos sanguíneos lo que permitirá darle solución óptima y rápida como exige el momento.

- Para las solicitudes electivas se le dará solución cuando haya sangre de su mismo grupo y factor, en caso de no haber disponibilidad de la misma no se le podrá dar solución a este pedido.

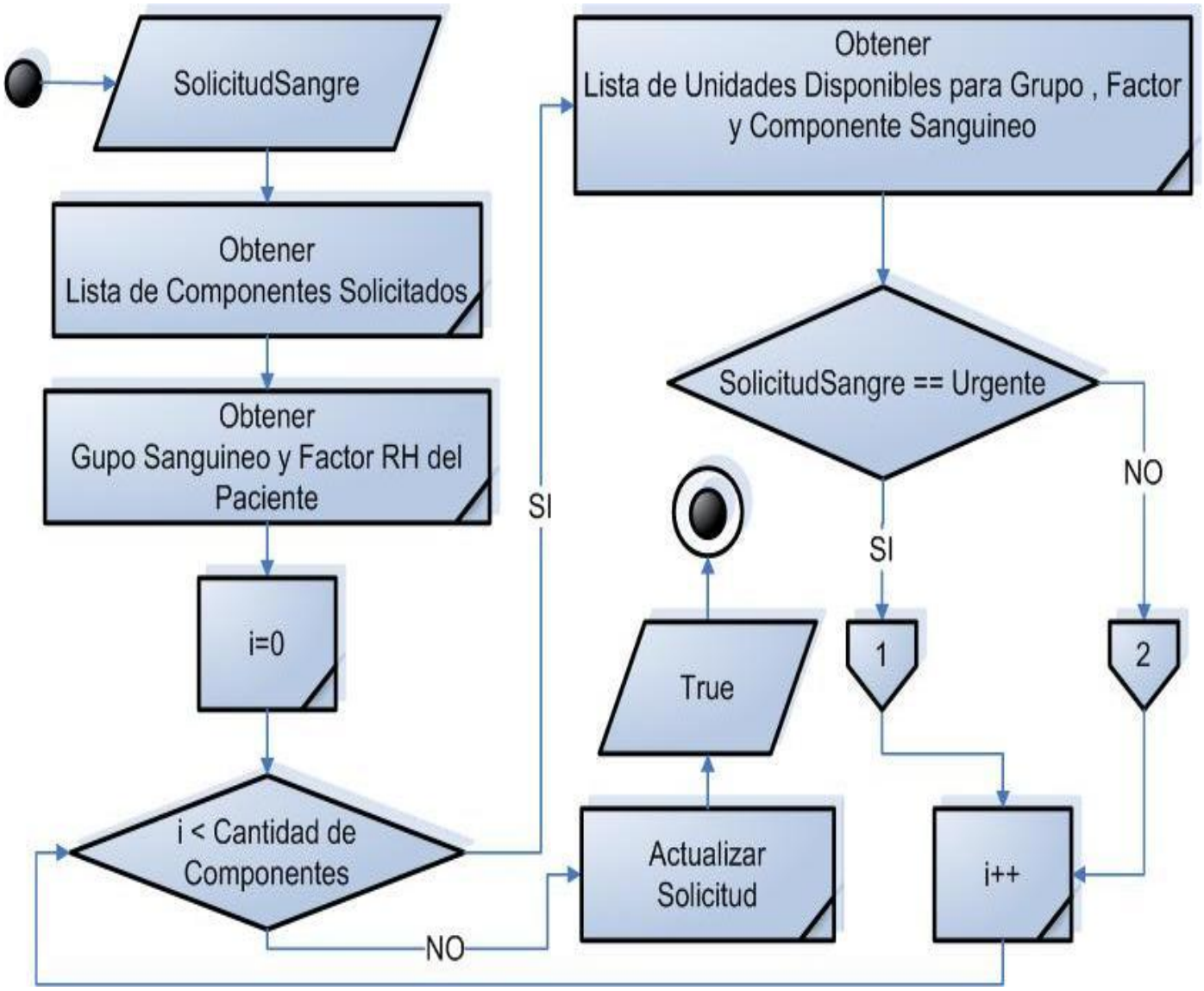

Fig. 5

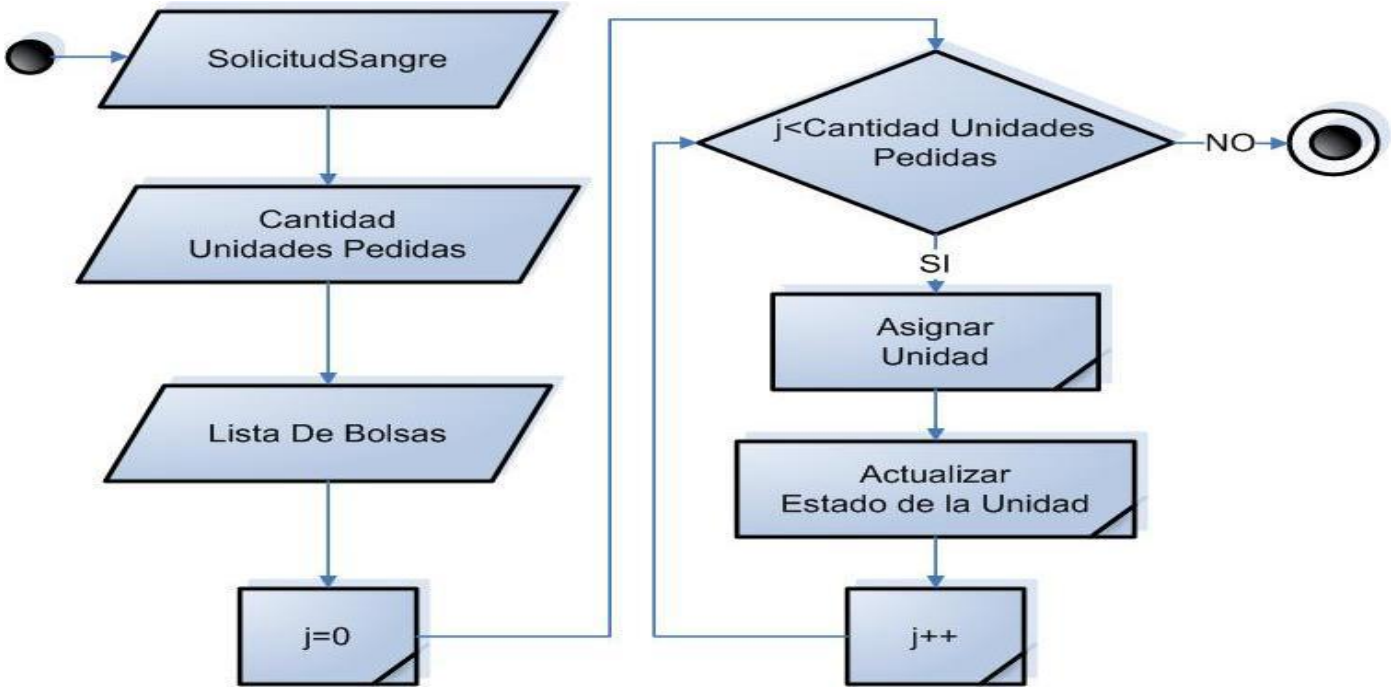
SolicitudSangre
Cantidad
Unidades Pedidas
Lista De Bolsas
j=0
j<Cantidad Unidades
Pedidas
NO
SI
Asignar
Unidad
Actualizar
Estado de la Unidad
j++

Fig. 6

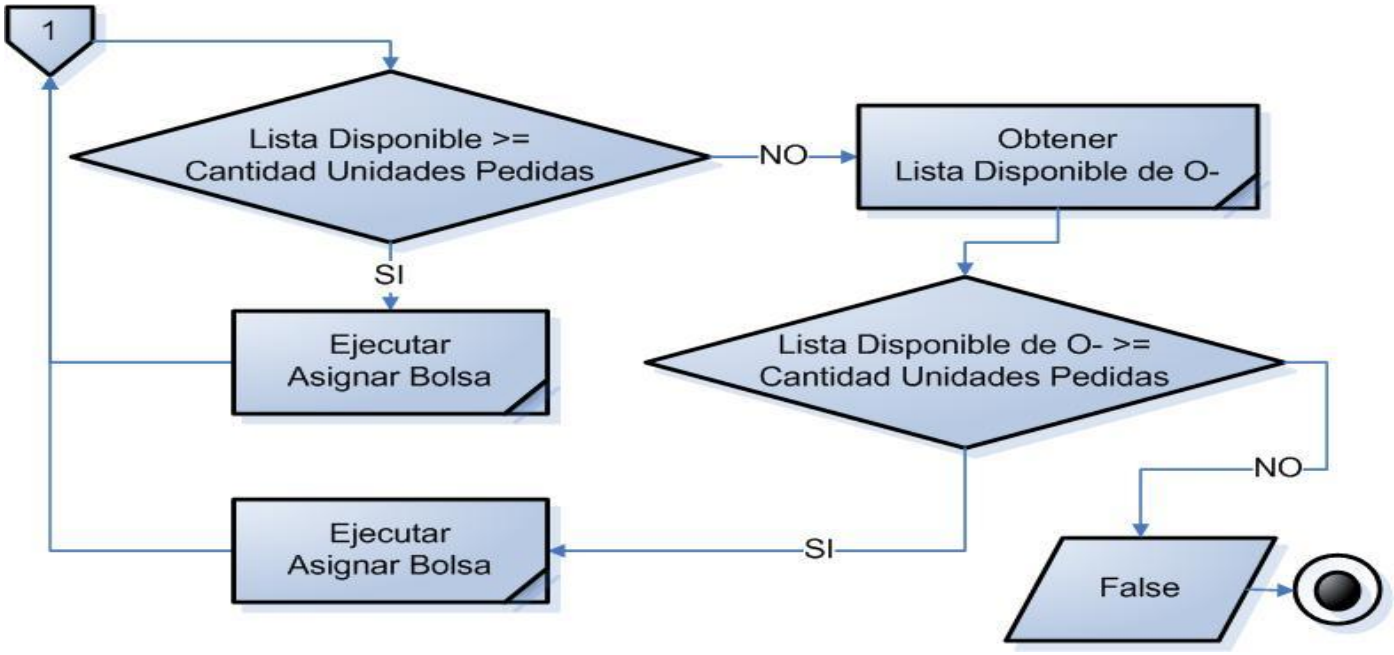
1
Lista Disponible >=
Cantidad Unidades Pedidas
NO
Obtener
Lista Disponible de O-
SI
Ejecutar
Asignar Bolsa
Lista Disponible de O- >=
Cantidad Unidades Pedidas
NO
Ejecutar
Asignar Bolsa
SI
False

Fig.7

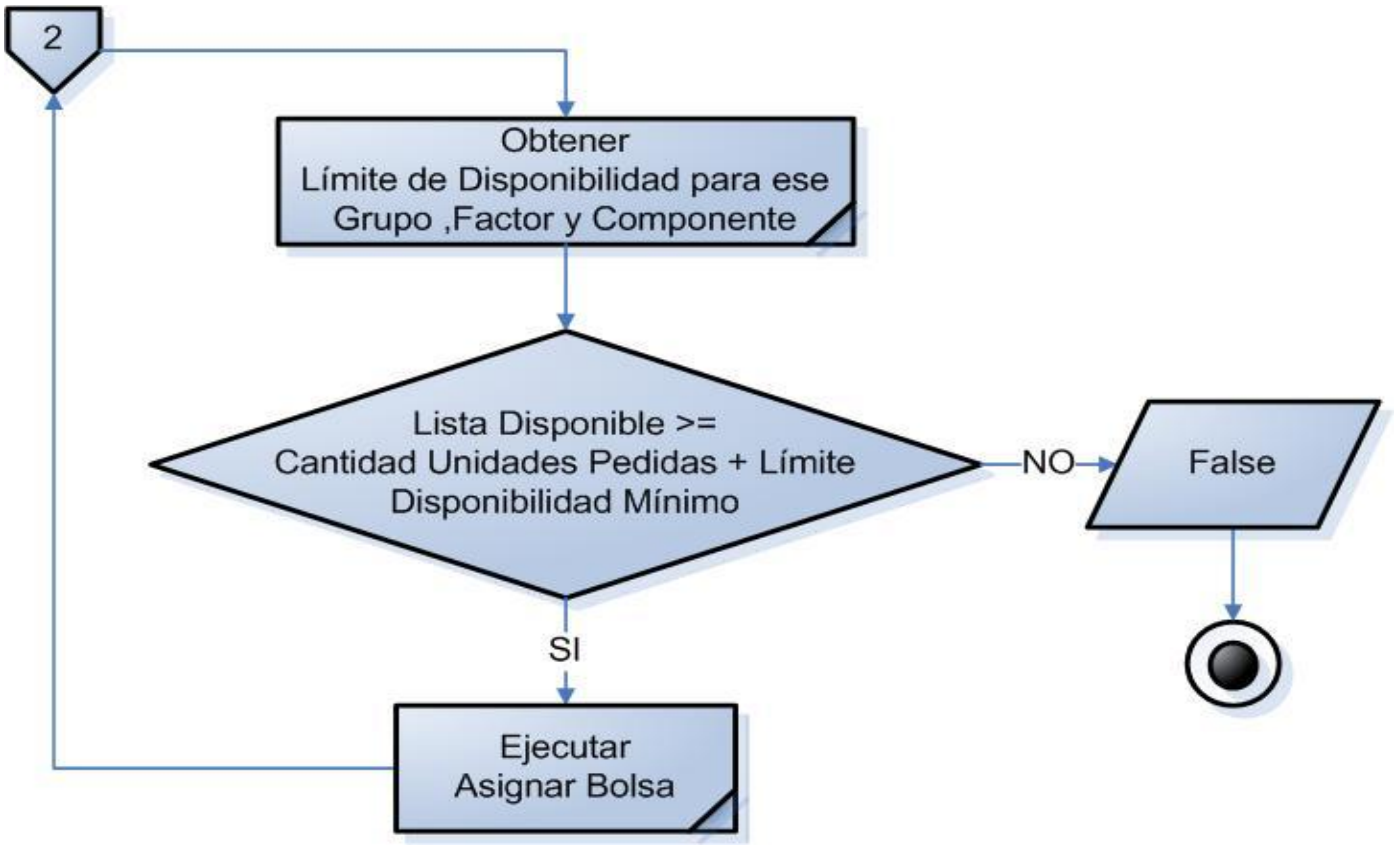

Fig.8

2.4.1 Análisis de la complejidad del algoritmo

La complejidad de un algoritmo se calcula a través del cálculo de la complejidad ciclomática del código del mismo. Primeramente se enumeran cada una de las líneas de código.

```
public bool AtenderSolicitud(SolicitudSangre solicitud)
        {
            List<ComponenteSanguineoBsSolicitud> componentesSolicitados = new
List<ComponenteSanguineoBsSolicitud>();  1
            ComponenteSanguineoBsSolicitud componentesSolicitadosParametro = new
ComponenteSanguineoBsSolicitud();  1
            componentesSolicitadosParametro.IdSolicitud = solicitud.IdSolicitud;
            ComponenteSanguineoBsSolicitudNegocio compSolNeg = new
ComponenteSanguineoBsSolicitudNegocio(); 1
```

```
componentesSolicitados =
compSolNeg.ObtenerTodos(componentesSolicitadosParametro);  1
GrupoSanguineoPersona grupoSanguineoPersona = new
GrupoSanguineoPersona();  1
GrupoSanguineoPersona grupoSanguineoPersonaParametro = new
GrupoSanguineoPersona();  1
GrupoSanguineoPersonaNegocio grupoSanguineoPersonaNegocio = new
GrupoSanguineoPersonaNegocio();  1
grupoSanguineoPersonaParametro.IdPersona = solicitud.IdPersona;
grupoSanguineoPersona =
grupoSanguineoPersonaNegocio.ObtenerUno(grupoSanguineoPersonaParametro);  1
BolsaView bolsaViewParametro = new BolsaView();  1
bolsaViewParametro.IdGrupoSanguineo =
grupoSanguineoPersona.IdGrupoSanguineo;  1
bolsaViewParametro.IdFactorRh = grupoSanguineoPersona.IdFactorRh;
bolsaViewParametro.EstadoBolsa = "Disponible"; 1

for (int i = 0; i < componentesSolicitados.Count; i++)  2
{
   List<BolsaView> listaDisponibles = new List<BolsaView>();  3
   bolsaViewParametro.IdComponente = componentesSolicitados[i].IdComponente;
3
   BolsaViewRepositorio bolsaViewRep = new BolsaViewRepositorio();  3
   listaDisponibles = bolsaViewRep.ObtenerTodos(bolsaViewParametro); 3
   if(solicitud.CaracterUrgente==true) 4
   {
      if (listaDisponibles.Count >= componentesSolicitados[i].Cantidad)  5
      {
         AsignarBolsa((int)componentesSolicitados[i].Cantidad, solicitud,
listaDisponibles);  6
      }
      else
```

```
                {
                    List<BolsaView> listaDisponiblesOnegativo = new List<BolsaView>(); 7
                    BolsaView bolsaViewOnegativo = new BolsaView();  7
                    bolsaViewOnegativo.IdComponente =
componentesSolicitados[i].IdComponente;  7
                    bolsaViewOnegativo.GrupoSanguineo = "O"; 7
                    bolsaViewOnegativo.FactorRh = "-"; 7
                    bolsaViewOnegativo.EstadoBolsa = "Disponible"; 7
                    listaDisponiblesOnegativo =
bolsaViewRep.ObtenerTodos(bolsaViewOnegativo); 7
                    if (listaDisponiblesOnegativo.Count >= componentesSolicitados[i].Cantidad)
8
                    {
                        AsignarBolsa((int)componentesSolicitados[i].Cantidad, solicitud,
listaDisponiblesOnegativo); 9
                    }
                    else
                    {
                      return false; 10
                    } 11
                }
            } 12
            else
            {
                LimiteDisponibilidadNegocio limiteDisponibilidadNeg = new
LimiteDisponibilidadNegocio(); 13
                LimiteDisponibilidad limiteDisponibilidad = new LimiteDisponibilidad();  13
                limiteDisponibilidad.IdComponente = componentesSolicitados[i].IdComponente;
13
                limiteDisponibilidad.IdGrupoSanguineo =
grupoSanguineoPersona.IdGrupoSanguineo;  13
                limiteDisponibilidad.IdFactorRh = grupoSanguineoPersona.IdFactorRh;  13
```

```
                limiteDisponibilidad =
limiteDisponibilidadNeg.ObtenerUno(limiteDisponibilidad);  13
                if (listaDisponibles.Count >= componentesSolicitados[i].Cantidad +
limiteDisponibilidad.CantMin) 14
                {
                    AsignarBolsa((int)componentesSolicitados[i].Cantidad, solicitud,
listaDisponibles);  15
                }
                else
                {
                    return false; 16
                }
            } 17

        } 18
        solicitud.SolicitudAtendida = true; 19
        Actualizar(solicitud);  19
            return true;  19
    } 20
```

Después de este paso, es necesario representar el grafo de flujo asociado, en el cual se representan distintos componentes como es el caso de:

Nodo: son los círculos representados en el grafo de Flujo, el cual representa una o más secuencias de procedimientos. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que existan nodos hallan nodos que no se asocien, estos nodos se utilizan principalmente al inicio y final del grafo.

Aristas: son constituidas por las flechas del grafo y representan el flujo de control, son iguales a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente la sentencia de un procedimiento.

Regiones: son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran y la cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

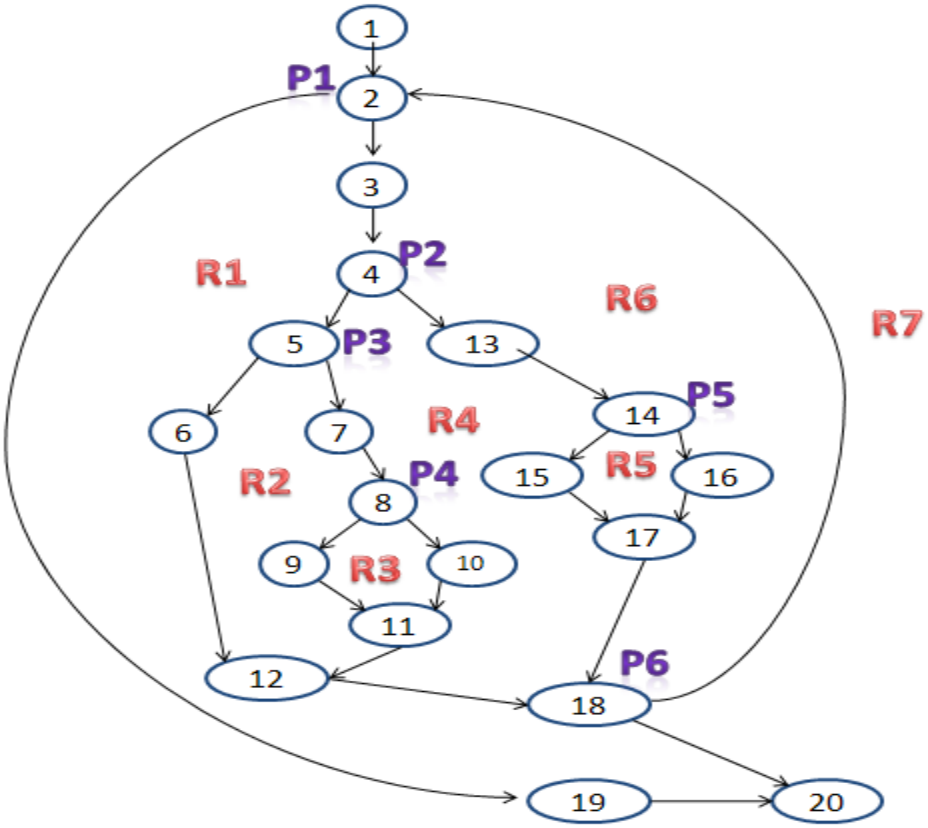

Fig. 9. Representación del Grafo de Flujo

Seguidamente a la construcción del grafo de flujo se procede a efectuar el cálculo de la complejidad ciclomática del código, el cálculo es necesario efectuarlo mediante tres vías formulas que para concluir con que el cálculo fue correcto es necesario que por las tres vías del mismo resultado, las fórmulas para calcular son las siguientes:

1- V (G) = (A – N) + 2

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

2- V (G) = P + 1

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

3- V (G) = R

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del calculo.

Calculo por la vía 1:

V (G) = (A – N) + 2

V (G) = (25– 20) + 2
V (G) = 7
Calculo por la vía 2:
V (G) = P + 1
V (G) = 6 + 1
V (G) = 7
Calculo por la vía 3:
V (G) = R
V (G) = 7

2.5 Selección de las estructuras de datos apropiadas para la implementación.

Para la implementación de los algoritmos no triviales, resultó necesario el uso del TDA Lista
Las listas son tipos de datos abstractos de amplio uso en computación.
Se puede definir el TDA lista. Matemáticamente, una lista es una secuencia de cero o más elementos de un tipo determinado (conocido como el tipo_de_elemento). A menudo, se representa una lista como una sucesión de elementos separados por comas:
A1, A2,..., An

donde n>=0
cada Ai es del tipo tipo_de_elemento
A1 es el primer elemento de la lista
An es el último elemento de la lista
Si n=0 se tiene una lista vacía, es decir, no tiene elementos

El TDA lista deriva de la noción matemática de lista, más una definición de operaciones sobre la misma. Típicamente, las operaciones sobre un TDA lista suelen ser: insertar elemento, eliminar elemento, localizar elemento, anular lista, etcétera. No se entrará en el detalle de la implementación las operaciones. La realización de los TDA lista, es posible mediante asignación de memoria estática, como vectores (arrays), o mediante la asignación de memoria dinámica y punteros.

Se hará uso de las listas porque toda la información devuelta por el repositorio será a través de este tipo de dato, tal es el caso de los listados de pacientes, listados de donantes, listado de bolsas de sangre, listados de solicitudes echas al Banco de Sangre, listado de componentes entre muchas otras.

2.6 Descripción de las nuevas clases u operaciones necesarias.

2.6.1 Clases Controladoras

Las clases de control coordinan el trabajo de uno o unos pocos casos de uso, coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso, por lo que definen el flujo de control y las transacciones dentro de un caso de uso delegando el trabajo a otros objetos.

Tabla 2.6.1: Bolsa Negocio

Nombre: BolsaNegocio	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
repositorio	BolsaRepositorio
Responsabilidades:	
Nombre:	**BolsaNegocio**
Descripción:	Constructor de la Clase.
Nombre	**ObtenerUno**(Bolsa entidad)
Descripción:	Método para obtener una Bolsa.
Nombre:	**ObtenerTodos**(Bolsa entidad)
Descripción:	Método para obtener una lista de Bolsas.
Nombre:	**ObtenerTodos**
Descripción:	Método para obtener una lista de Bolsas.
Nombre:	**VistaObtenerTodos**(BolsaView entidad)
Descripción:	Método para obtener una lista de Bolsas de la vista que contiene todas las descripciones de las bolsas disponibles y consumidas.
Nombre:	**Adicionar**(Bolsa entidad)
Descripción:	Método para agregar una Bolsa.

Nombre:	**ActualizarEstado**(Bolsa entidad)
Descripción:	Método para actualizar el estado de una Bolsa.
Nombre:	**Eliminar**(Bolsa entidad, CausaSalidaBolsa causa)
Descripción:	Método para Eliminar una Bosa. Además actualiza el estado de la Bolsa según el motivo de la salida.
Nombre:	**DevolverBolsa**(Bolsa entidad)
Descripción:	Método para devolver una Bolsa que se le haya dado salida para transfusión y no se haya utilizado. Además actualiza el estado de la Bolsa a “Disponible”.

Tabla 2.6.2 Bolsa Repositorio

Nombre: BolsaRepositorio : Repositorio<Bolsa>	
Tipo de clase: Controladora	
Atributo :	**Tipo:**
factory	IDomainObjectFactory<Bolsa>
Responsabilidades :	
Nombre:	**BolsaRepositorio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno**(Bolsa entity)
Descripción:	Método para obtener una Bolsa.
Nombre:	**ObtenerTodos**
Descripción:	Método para obtener todas las Bolsas.
Nombre:	**ObtenerTodos**(Bolsa entity)
Descripción:	Método para obtener una lista de Bolsas.
Nombre:	**Adicionar**(ref Bolsa entity)
Descripción:	Método para agregar una Bolsa.
Nombre:	**Adicionar**(IDbTransaction transaction, ref Bolsa entity)
Descripción:	Método para agregar una Bolsa, con transacción.
Nombre:	**Actualizar**(Bolsa entity)
Descripción:	Método para actualizar el estado de una Bolsa.

Nombre:	**Actualizar**(IDbTransaction transaction,Bolsa entity)
Descripción:	Método para actualizar una Bolsa, con transacción.
Nombre:	**Eliminar**(Bolsa entity)
Descripción:	Método para eliminar una Bolsa.
Nombre:	**Eliminar**(IDbTransaction transaction,Bolsa entity)
Descripción:	Método para eliminar una Bolsa, con transacción.

Tabla 2.6.3 Componente Sanguíneo Negocio

Nombre: ComponenteSanguineoNegocio	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
repositorio	ComponenteSanguineoRepositorio
Responsabilidades:	
Nombre:	**ComponenteSanguineoNegocio**
Descripión:	Constructor de la Clase.
Nombre:	**ObtenerUno**(ComponenteSanguineo entidad)
Descripión:	Método para obtener un Componente Sanguineo.
Nombre:	**ObtenerTodos**(ComponenteSanguineo entidad)
Descripión:	Método para obtener una lista de Componentes Sanguíneos.
Nombre:	**ObtenerTodos**
Descripión:	Método para obtener todos los Componentes Sanguíneos.
Nombre:	**Adicionar**(ComponenteSanguineo entidad)
Descripión:	Método para agregar un Componente Sanguíneo.
Nombre:	**Actualizar**(ComponenteSanguineo entidad)
Descripión:	Método para actualizar un Componente Sanguíneo.
Nombre:	**Eliminar**(ComponenteSanguineo entidad)
Descripión:	Método para eliminar un Componente Sanguíneo.

Tabla 2.6.4 Componente Sanguíneo Repositorio

Nombre: ComponenteSanguineoRepositorio : Repositorio<ComponenteSanguineo>	
Tipo de clase: Controladora	
Atributo :	**Tipo:**
factory	IDomainObjectFactory<ComponenteSanguineo>
Responsabilidades :	
Nombre:	**ComponenteSanguineoRepositorio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno**(ComponenteSanguineo entity)
Descripción:	Método para obtener un Componente Sanguíneo.
Nombre:	**ObtenerTodos**
Descripción:	Método para obtener todos los Componentes Sanguíneos.
Nombre:	**ObtenerTodos**(ComponenteSanguineo entity)
Descripción:	Método para obtener una lista de Componentes Sanguíneos.
Nombre:	**Adicionar**(ref ComponenteSanguineo entity)
Descripción:	Método para agregar un Componente Sanguíneo.
Nombre:	**Adicionar**(IDbTransactiontransaction, ref ComponenteSanguineo entity)
Descripción:	Método para agregar un Componente Sanguíneo, con transacción.
Nombre:	**Actualizar**(ComponenteSanguineo entity)
Descripción:	Método para actualizar un Componente Sanguíneo.
Nombre:	**Actualizar**(IDbTransaction transaction, ComponenteSanguineo entity)
Descripción:	Método para actualizar un Componente Sanguíneo, con transacción
Nombre:	**Eliminar**(ComponenteSanguineo entity)
Descripción:	Método para eliminar un Componente Sanguíneo.
Nombre:	**Eliminar**(IDbTransaction transaction,ComponenteSanguineo entity
Descripción:	Método para eliminar un Componente Sanguíneo, con transacción.

Tabla 2.6.5 Ficha donación Negocio

Nombre: FichaDonacionNegocio	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
fichaDonacionRep	FichaDonacionRepositorio
pruebaLaboratorioRep	PruebaLaboratorioRepositorio
examenFisicoDonanteRep	ExamenFisicoDonanteRepositorio
fichaDonacionEnfermedadRep	FichaDonacionEnfermedadRepositorio
Responsabilidades:	
Nombre:	**FichaDonacionNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno**(FichaDonacion entidad)
Descripción:	Método para obtener una Ficha de Donación.
Nombre:	**ObtenerTodos**(FichaDonacion entidad)
Descripción:	Método para obtener una lista de Fichas de Donaciones.
Nombre:	**FichaDonacionObtenerTodos**
Descripción:	Método para obtener todos las Fichas Donaciones.
Nombre:	**Adicionar**(FichaDonacion entidad)
Descripción:	Método para agregar una Ficha de Donación.
Nombre:	**Adicionar**(FichaDonacion fichaDonacion, ExamenFisicoDonante examenFisico, List<Diagnostico> listaEnfermedades)
Descripción:	Método para adicionar una Ficha de Donación completa.
Nombre:	**Actualizar**(FichaDonacion entidad)
Descripción:	Método para actualizar una Ficha de Donación.
Nombre:	**Eliminar**(FichaDonacion entidad)
Descripción:	Método para eliminar una Ficha de Donación.
Nombre:	**ObtenerUno**(PruebaLaboratorio entidad)
Descripción:	Método para obtener una Prueba de Laboratorio.
Nombre:	**ObtenerTodos**(PruebaLaboratorio entidad)

Descripción:	Método para obtener una lista de Pruebas de Laboratorio.
Nombre:	**PruebaLaboratorioObtenerTodos**
Descripción:	Método para obtener todas las Pruebas de Laboratorio.
Nombre:	**Adicionar**(PruebaLaboratorio entidad)
Descripción:	Método para adicionar una Prueba de Laboratorio.
Nombre:	**Actualizar**(PruebaLaboratorio entidad)
Descripción:	Método para actualizar una Prueba de Laboratorio.
Nombre:	**Eliminar**(PruebaLaboratorio entidad)
Descripción:	Método para eliminar una Prueba de Laboratorio.
Nombre:	**ObtenerUno**(ExamenFisicoDonante entidad)
Descripción:	Método para obtener un Examen Físico.
Nombre:	**ObtenerTodos**(ExamenFisicoDonante entidad)
Descripción:	Método para obtener una lista de Exámenes Físicos.
Nombre:	**ExamenFisicoObtenerTodos**
Descripción:	Método para obtener todos los Exámenes Físicos.
Nombre:	**Adicionar**(ExamenFisicoDonante entidad)
Descripción:	Método para adicionar un Examen Físico.
Nombre:	**Actualizar**(ExamenFisicoDonante entidad)
Descripción:	Método para actualizar un Examen Físico.
Nombre:	**Eliminar**(ExamenFisicoDonante entidad)
Descripción:	Método para eliminar un Examen Físico.
Nombre:	**ObtenerUno**(FichaDonacionEnfermedad entidad)
Descripción:	Método para obtener una Enfermedad correspondiente a una Ficha de Donación.
Nombre:	**ObtenerTodos**(FichaDonacionEnfermedad entidad)
Descripción:	Método para obtener la lista de enfermedades correspondientes a una Ficha de Donación.
Nombre:	**EnfermedadObtenerTodos**
Descripción:	Método para obtener todas las enfermedades.
Nombre:	**Adicionar**(FichaDonacionEnfermedad entidad)

Descripción:	Método para adicionar una Enfermedad correspondiente a una Ficha de Donación.
Nombre:	**Actualizar**(FichaDonacionEnfermedad entidad)
Descripción:	Método para actualizar una Enfermedad correspondiente a una Ficha de Donación.
Nombre:	**Eliminar**(FichaDonacionEnfermedad entidad)
Descripción:	Método para eliminar una Enfermedad correspondiente a una Ficha de Donación.
Nombre:	**FichaSinExamenLaboratorio**(FichaDonacion fichaDonacion)
Descripción:	Método para obtener una lista de las Fichas de Donaciones sin Examen de Laboratorio.
Nombre:	**ObtenerUno**(UltimaFicha ultimaFicha)
Descripción:	Método para obtener la ultima Ficha de Donación insertada en la base de datos.

Tabla 2.6.6 Solicitud Sangre Negocio

Nombre: SolicitudSangreNegocio	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
repositorio	SolicitudSangreRepositorio
Responsabilidades:	
Nombre:	**SolicitudSangreNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno**(SolicitudSangre entidad)
Descripción:	Método para obtener una Solicitud de Sangre.
Nombre:	**ObtenerTodos**(SolicitudSangre entidad)
Descripción:	Método para obtener una lista de Solicitudes de Sangre.
Nombre:	**ObtenerTodos**
Descripción:	Método para obtener todas las Solicitudes de Sangre.

Nombre:	**Adicionar**(SolicitudSangre entidad)
Descripción:	Método para adicionar una Solicitud de Sangre.
Nombre:	**Actualizar**(SolicitudSangre entidad)
Descripción:	Método para actualizar una Solicitud de Sangre.
Nombre:	**Eliminar**(SolicitudSangre entidad)
Descripción:	Método para eliminar una Solicitud de Sangre.
Nombre:	**AtenderSolicitud**(SolicitudSangre solicitud)
Descripción:	Método para dar solución a una Solicitud de Sangre hecha al banco de sangre.
Nombre:	**AsignarBolsa**(int cantidadUnidades, SolicitudSangre solicitud, List<BolsaView> listaDeBolsas)
Descripción:	Método para asignar las Unidades("Bolsas") a una Solicitud hecha al Banco de Sangre.

Tabla 2.6.7 Grupo Sanguíneo Persona Negocio

Nombre: GrupoSanguineoPersonaNegocio	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
repositorio	GrupoSanguineoPersonaRepositorio
Responsabilidades:	
Nombre:	**GrupoSanguineoPersonaNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno**(GrupoSanguineoPersona entidad)
Descripción:	Método para obtener el Grupo Sanguíneo de una Persona.
Nombre:	**ObtenerTodos**(GrupoSanguineoPersona entidad)
Descripción:	Método para obtener una lista de las personas con determinado Grupo Sanguíneo.
Nombre:	**ObtenerTodos**
Descripción:	Método para obtener todas las personas que tienen registrado su

	Grupo Sanguíneo.
Nombre:	**Adicionar**(GrupoSanguineoPersona entidad)
Descripción:	Método para registrarle el Grupo Sanguíneo a una Persona.
Nombre:	**Actualizar**(GrupoSanguineoPersona entidad)
Descripción:	Método para actualizarle el Grupo Sanguíneo a una Persona.
Nombre:	**Eliminar**(GrupoSanguineoPersona entidad)
Descripción:	Método para eliminarle el Grupo Sanguíneo a una Persona.

Tabla 2.6.8 Prueba Prestansfusional Negocio

Nombre: PruebaPretransfuncionalNegocio	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
repositorio	PruebaPretransfuncionalRepositorio
Responsabilidades:	
Nombre:	**PruebaPretransfuncionalNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno**(PruebaPretransfuncional entidad)
Descripción:	Método para obtener una Prueba Pretransfuncional.
Nombre:	**ObtenerTodos**(PruebaPretransfuncional entidad)
Descripción:	Método para obtener una lista de Prueba Pretransfuncionales.
Nombre:	**ObtenerTodos**
Descripción:	Método para obtener todas las Pruebas Pretransfuncionales.
Nombre:	**Adicionar**(PruebaPretransfuncional entidad)
Descripción:	Método para adicionar una Prueba Pretransfuncional.
Nombre:	**Actualizar**(PruebaPretransfuncional entidad)
Descripción:	Método para actualizar una Prueba Pretransfuncional.
Nombre:	**Eliminar**(PruebaPretransfuncional entidad)
Descripción:	Método para eliminar una Prueba Pretransfuncional.

2.6.2 Clases Entidad

Estas clases modelan información que poseen una larga vida y que a menudo son conceptos y sucesos que ocurren en el mundo real. La fuente principal de obtención son las clases entidades del negocio y el glosario de términos que se ha ido elaborando.
Se encargan de modelar la información del sistema y el comportamiento asociado a una información.

Tabla 2.6.9 Ficha Donación

Nombre: FichaDonacion	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
nroFicha	Int32
fecha	DateTime
idPersona	Int32
idTipoDonante	Int32
unidad	Int32
hemoglobina	Double
apto	Boolean
cantSangredonada	Int32
viajesExterior	Boolean
observaciones	String
idFuncionario	Int32
Responsabilidades:	
Nombre:	**FichaDonacion**
Descripción:	Constructor de la Clase.
Nombre:	**NroFicha**
Descripción:	Propiedad para mostrar y modificar el NroFicha.
Nombre:	**Fecha**
Descripción:	Propiedad para mostrar y modificar la Fecha.
Nombre:	**IdPersona**
Descripción:	Propiedad para mostrar y modificar el IdPersona.

Nombre:	**IdTipoDonante**
Descripción:	Propiedad para mostrar y modificar el IdTipoDonante.
Nombre:	**Unidad**
Descripción:	Propiedad para mostrar y modificar la Unidad.
Nombre:	**Hemoglobina**
Descripción:	Propiedad para mostrar y modificar la Hemoglobina.

Tabla 2.6.10 Examen Físico Donante

Nombre: ExamenFisicoDonante	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
nroFicha	Int32
fecha	DateTime
pulsoDonante	Int32
pesoDonante	Double
temperaturaDonante	Double
tensionArterialMax	Int32
tensionArterialMin	Int32
Responsabilidades:	
Nombre:	**ExamenFisicoDonante**
Descripción:	Constructor de la Clase.
Nombre:	**NroFicha**
Descripción:	Propiedad para mostrar y modificar el NroFicha.
Nombre:	**Fecha**
Descripción:	Propiedad para mostrar y modificar la Fecha.
Nombre:	**PulsoDonante**
Descripción:	Propiedad para mostrar y modificar el PulsoDonante.
Nombre:	**PesoDonante**

Descripción:	Propiedad para mostrar y modificar el PesoDonante.
Nombre:	**TemperaturaDonante**
Descripción:	Propiedad para mostrar y modificar la TemperaturaDonante.
Nombre:	**TensionArterialMax**
Descripción:	Propiedad para mostrar y modificar la TensionArterialMax.
Nombre:	**TensionArterialMin**
Descripción:	Propiedad para mostrar y modificar la TensionArterialMin.

Tabla 2.6.11 Prueba Laboratorio

Nombre: PruebaLaboratorio	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
nroFicha	Int32
fecha	DateTime
anticuerposAtipicos	Boolean
vdrl	Boolean
antigenoSuperficie	Boolean
hepatitisB	Boolean
vih	Boolean
otras	Boolean
idFuncionario	Int32
Responsabilidades:	
Nombre:	**PruebaLaboratorio**
Descripción:	Constructor de la Clase.
Nombre:	**NroFicha**
Descripción:	Propiedad para mostrar y modificar el NroFicha.
Nombre:	**Fecha**
Descripción:	Propiedad para mostrar y modificar la Fecha.

Nombre:	**AnticuerposAtipicos**
Descripción:	Propiedad para mostrar y modificar los AnticuerposAtipicos.
Nombre:	**Vdrl**
Descripción:	Propiedad para mostrar y modificar el Vdrl.
Nombre:	**AntigenoSuperficie**
Descripción:	Propiedad para mostrar y modificar los AntigenoSuperficie.
Nombre:	**HepatitisB**
Descripción:	Propiedad para mostrar y modificar la HepatitisB.
Nombre:	**Vih**
Descripción:	Propiedad para mostrar y modificar el Vih.
Nombre:	**Otras**
Descripción:	Propiedad para mostrar y modificar Otras.
Nombre:	**IdFuncionario**
Descripción:	Propiedad para mostrar y modificar el IdFuncionario.

Tabla 2.6.12 Bolsa

Nombre: Bolsa	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
nroBolsa	String
idGrupoSanguineo	Int32
idComponente	Int32
idEstado	Int32
idEntrada	Int32
idFactorRh	Int32
cantidadMl	Int32
fechaVencimiento	DateTime
Responsabilidades:	

Nombre:	**Bolsa**
Descripción:	Constructor de la Clase.
Nombre:	**NroBolsa**
Descripción:	Propiedad para mostrar y modificar el NroBolsa.
Nombre:	**IdGrupoSanguineo**
Descripción:	Propiedad para mostrar y modificar el IdGrupoSanguineo.
Nombre:	**IdComponente**
Descripción:	Propiedad para mostrar y modificar el IdComponente.
Nombre:	**IdEstado**
Descripción:	Propiedad para mostrar y modificar el IdEstado.
Nombre:	**IdEntrada**
Descripción:	Propiedad para mostrar y modificar el IdEntrada.
Nombre:	**IdFactorRh**
Descripción:	Propiedad para mostrar y modificar el IdFactorRh.
Nombre:	**CantidadMl**
Descripción:	Propiedad para mostrar y modificar la CantidadMl.
Nombre:	**FechaVencimiento**
Descripción:	Propiedad para mostrar y modificar la FechaVencimiento.

Tabla 2.6.13 Componente sanguíneo

Nombre: ComponenteSanguineo	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
idComponente	Int32
descripción	String
Responsabilidades:	
Nombre:	**ComponenteSanguineo**
Descripción:	Constructor de la Clase.

Nombre:	**IdComponente**
Descripción:	Propiedad para mostrar y modificar el IdComponente.
Nombre:	**DESCRIPCIÓN**
Descripción:	Propiedad para mostrar y modificar la Descripción.

Tabla 2.6.14 Solicitud de Sangre

Nombre: SolicitudSangre	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
idSolicitud	Int32
fechaSolicitud	DateTime
caracterUrgente	Boolean
solicitudAtendida	Boolean
sala	String
cama	String
salonOperaciones	Boolean
pruebasCompatibilidad	Boolean
idPersona	Int32
idDiagnostico	String
transfusion	Boolean
Responsabilidades:	
Nombre:	**SolicitudSangre**
Descripción:	Constructor de la Clase.
Nombre:	**IdSolicitud**
Descripción:	Propiedad para mostrar y modificar el IdSolicitud.
Nombre:	**FechaSolicitud**
Descripción:	Propiedad para mostrar y modificar la FechaSolicitud.
Nombre:	**CaracterUrgente**

Descripción:	Propiedad para mostrar y modificar el CaracterUrgente.
Nombre:	**SolicitudAtendida**
Descripción:	Propiedad para mostrar y modificar la SolicitudAtendida.
Nombre:	**Sala**
Descripción:	Propiedad para mostrar y modificar la Sala.
Nombre:	**Cama**
Descripción:	Propiedad para mostrar y modificar la Cama.
Nombre:	**SalonOperaciones**
Descripción:	Propiedad para mostrar y modificar el SalonOperaciones.
Nombre:	**PruebasCompatibilidad**
Descripción:	Propiedad para mostrar y modificar la PruebasCompatibilidad.
Nombre:	**IdPersona**
Descripción:	Propiedad para mostrar y modificar el IdPersona.
Nombre:	**IdDiagnostico**
Descripción:	Propiedad para mostrar y modificar el IdDiagnostico.
Nombre:	**Transfusion**
Descripción:	Propiedad para mostrar y modificar la Transfusion.

Tabla 2.6.15 Componente Sanguíneo

Nombre: ComponenteSanguineoBsSolicitud	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
idComponente	Int32
idSolicitud	Int32
cantidad	Int32
Responsabilidades:	
Nombre:	**ComponenteSanguineoBsSolicitud**
Descripción:	Constructor de la Clase.

Nombre:	**IdComponente**
Descripción:	Propiedad para mostrar y modificar el IdComponente.
Nombre:	**IdSolicitud**
Descripción:	Propiedad para mostrar y modificar el IdSolicitud.
Nombre:	**Cantidad**
Descripción:	Propiedad para mostrar y modificar la Cantidad.

Tabla 2.6.16 Grupo Sanguíneo

Nombre: GrupoSanguineoPersona	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
idPersona	Int32
idGrupoSanguineo	Int32
idFactorRh	Int32
Responsabilidades:	
Nombre:	**GrupoSanguineoPersona**
Descripción:	Constructor de la Clase.
Nombre:	**IdPersona**
Descripción:	Propiedad para mostrar y modificar el IdPersona.
Nombre:	**IdGrupoSanguineo**
Descripción:	Propiedad para mostrar y modificar el IdGrupoSanguineo.
Nombre:	**IdFactorRh**
Descripción:	Propiedad para mostrar y modificar el IdFactorRh.

Tabla 2.6.17 Prueba Pretransfusionales

Nombre: PruebaPretransfuncional	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
idPersona	Int32
fecha	DateTime
hemoglobina	Double
hematiesHematocristo	Double
idFuncionario	Int32
Responsabilidades:	
Nombre:	**PruebaPretransfuncional**
Descripción:	Constructor de la Clase.
Nombre:	**IdPersona**
Descripción:	Propiedad para mostrar y modificar el IdPersona.
Nombre:	**Fecha**
Descripción:	Propiedad para mostrar y modificar la Fecha.
Nombre:	**Hemoglobina**
Descripción:	Propiedad para mostrar y modificar la Hemoglobina.
Nombre:	**HematiesHematocristo**
Descripción:	Propiedad para mostrar y modificar el HematiesHematocristo.
Nombre:	**IdFuncionario**
Descripción:	Propiedad para mostrar y modificar el IdFuncionario.

2.6.3 Clases Interfaz

Estas clases se encargan de modelar la interacción Actor-Sistema.

Son identificadas para cada interacción Actor-Caso de Uso, para cada sistema externo y para cada dispositivo que se utilice.

Tabla 2.6.18 Actualizar Ficha Donación

Nombre: Actualizar Ficha Donación	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
TextBox Fecha	TextBox
TextBox Peso	TextBox
TextBox Pulso	TextBox
TextBox Hemoglobina	TextBox
TextBox Temperatura	TextBox
TextBox Tension Arterial Max	TextBox
TextBox Tension Arterial Min	TextBox
TextBox Cantidad Sangre Donada	TextBox
DropD WonList AntigenoSupHB	DropD WonList
DropD WonList VIH	DropD WonList
DropD WonList Anticuerpos Atipicos	DropD WonList
DropD WonList VDRL	DropD WonList
CeckBox Apto	CeckBox
TextBox Observaciones	TextBox
TextBox Funcionario	TextBox
Button Enviar	Button
Button Cancelar	Button
Button Buscar Donante	Button
Responsabilidades:	
Nombre:	Cargar Pagina
Descripción:	Cargar todos los datos del donante.
Nombre:	Botón Actualizar
Descripción:	Actualizar los datos actualizables del donante en caso de algún cambio.
Nombre:	Botón Cancelar

Tabla 2.6.19 Registrar Ficha Donación.

Nombre: Registrar Ficha Donación	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
TextBox Fecha	TextBox
TextBox Peso	TextBox
TextBox Pulso	TextBox
TextBox Hemoglobina	TextBox
TextBox Temperatura	TextBox
TextBox Tension Arterial Max	TextBox
TextBox Tension Arterial Min	TextBox
TextBox Cantidad Sangre Donada	TextBox
DropD WonList AntigenoSupHB	DropD WonList
DropD WonList VIH	DropD WonList
DropD WonList Anticuerpos Atipicos	DropD WonList
DropD WonList VDRL	DropD WonList
CeckBox Apto	CeckBox
TextBox Observaciones	TextBox
TextBox Funcionario	TextBox
Button Enviar	Button
Button Cancelar	Button
Button Buscar Donante	Button
Responsabilidades:	
Nombre:	Button Enviar _Click
Descripción:	Enviar los datos de la ficha de donación de donante.
Nombre:	Button Cancelar_ Click
Descripción:	Cancelar la ficha de donación.
Nombre:	Button Buscar Donante Click
Descripción:	Buscar un donante para registrarle una donación.

Tabla 2.6.20 Adicionar Unidades

Nombre: Adicionar Unidades	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
DropD WonList Via Entrada	DropD WonList
TextBox Nro Bolsa	TextBox
DropD WonList Grupo Sanguíneo	DropD WonList
DropD WonList Factor RH	DropD WonList
DropD WonList Tipo Componente	DropD WonList
TextBox Capacidad Bolsa	TextBox
TextBox Fecha Vencimiento	TextBox
Button Adicionar	Button
Button Cancelar	Button
Responsabilidades:	
Nombre:	Cargar pagina
Descripción:	Cargar los datos para seleccionar las características de la Unidad a adicionar.
Nombre:	Boton Adicionar
Descripción:	Adicionar la unidad de sangre.

Tabla 2.6.21 Salida de componentes

Nombre: Salida de componentes	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
TextBox Nro Bolsa	TextBox
TextBox Tipo Componente	TextBox
TextBox Via Entrada	TextBox
DropDownList Causa Salida	DropDownList
Button Eliminar	Button

Button Cancelar	Button
Button Buscar Bolsa	Button
Responsabilidades:	
Nombre:	Cargar Pagina
Descripción:	Cargar los datos para seleccionar las causa de salida de la unidad
Nombre:	Eliminar
Descripción:	Eliminar la unidad.
Nombre:	Buscar Bolsa
Descripción:	Buscar la Bolsa (Unidad) que se desea eliminar.

Tabla 2.6.22 Buscar Bolsa

Nombre: Buscar Bolsa	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
Text Box NroBuscar	Text Box
Drop Down List Componente Buscar	Drop Down List
Text Box Fecha Buscar	Text Box
Drop Down List Grupo Sanguíneo	Drop Down List
Drop Down List Factor RH	Drop Down List
Drop Down List Via Entrada	Drop Down List
Button Buscar	Button
Button Seleccionar	Button
Responsabilidades:	
Nombre:	Cargar Pagina
Descripción:	Cargar los datos para realizar la búsqueda.
Nombre:	Buscar_Click
Descripción:	Buscar la bolsa según los parámetros seleccionados.
Nombre:	Seleccionar_Click
Descripción:	Seleccionar la unidad (Bolsa) que se desea.

Tabla 2.6.23 Configurar Stock

Nombre: Configurar Stock	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
Drop Down Componente Sanguíneo	Drop Down List
TextBox Stock Minimo	TextBox
TextBox Stock Maximo	TextBox
Button Aceptar	Button

Button Cancelar	Button
Responsabilidades:	
Nombre:	**Aceptar**
Descripción:	Enviar los limites del Stock
Nombre:	**Cancelar**
Descripción:	Cancelar en envío de los datos.

Tabla 2.6.24 Registrar Solicitud

Nombre: Registrar Solicitud	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
Text Box NombrePersona	Text Box
Text Box Primer Apellido	Text Box
Text Box Segundo Apellido	Text Box
Text Box Grupo Sanguíneo	Text Box
Text Box Factor RH	Text Box
Text Box Fecha Solicitud	Text Box
Text Box Diagnostico	Text Box
Text Box Cama	Text Box
Text Box Sala	Text Box
Text Box Conc Plaquetas	Text Box
Text Box PlasmaRico_enPlaquetas	Text Box
Text Box PlasmaFrescoCongelado	Text Box
Text Box Globulos Rojos	Text Box
Text Box Sangre Total	Text Box
Text Box Plasma	Text Box
Checkbox SalonOperaciones	Checkbox
Checkbox PruebasCompatibilidad	Checkbox

Checkbox Urgencia	Checkbox
Button Enviar Solicitud	Button
Button Buscar Paciente	Button
Responsabilidades:	
Nombre:	Enviar Solicitud
Descripción:	Enviar los datos de la solicitud.
Nombre:	Buscar Paciente
Descripción:	Buscar el paciente del cual se va realizar la solicitud.
Nombre:	Cargar Pagina
Descripción:	Cargar los datos que son determinables desde la base de datos.

Tabla 2.6.25 Cancelar Solicitud

Nombre: Cancelar Solicitud	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
Text Box NombrePersona	Text Box
Text Box Primer Apellido	Text Box
Text Box Segundo Apellido	Text Box
Text Box Grupo Sanguíneo	Text Box
Text Box Factor RH	Text Box
Text Box Fecha Solicitud	Text Box
Text Box Diagnostico	Text Box
Text Box Cama	Text Box
Text Box Sala	Text Box
Text Box Conc Plaquetas	Text Box
Text Box PlasmaRico_enPlaquetas	Text Box
Text Box PlasmaFrescoCongelado	Text Box
Text Box Globulos Rojos	Text Box

Text Box Sangre Total	Text Box
Text Box Plasma	Text Box
Checkbox SalonOperaciones	Checkbox
Checkbox PruebasCompatibilidad	Checkbox
Checkbox Urgencia	Checkbox
Button Enviar Solicitud	Button
Button Buscar Paciente	Button

Responsabilidades:	
Nombre:	Cargar Pagina
Descripción:	Cargar los tipos de factor y grupo sanguíneo.
Nombre:	Registrar GS_Click
Descripción:	Registrar el grupo sanguíneo y factor RH para ese paciente.
Nombre:	BuscarPaciente_Click
Descripción:	Buscar el paciente al cual se le va a registrar el grupo sanguino y factor RH

Tabla 2.6.26 Registrar pruebas pretransfusionales

Nombre: Registrar Pruebas PreTranfusionales	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
TextBoxNombre	TextBox
TextBoxPrimerApellido	TextBox
TextBoxSEgundoApellido	TextBox
TextBoxHemoglobina	TextBox
TextBoxHematiesHto	TextBox
ButtonRegistrarPruebas	Button
ButtonCancelar	Button
ButtonBuscarPaciente	Button
Responsabilidades:	
Nombre:	Cargar Paginas
Descripción:	Cargar los datos para las pruebas.
Nombre:	Registrar Prueba
Descripción:	Registrar las Pruebas.

2.6.4 Integración de las clases

La capa de presentación se encuentra compuesta por las clases de interfaz, las clases controladoras y las entidades. Las clases de interfaz se relacionan con las clases controladoras del negocio y ambas a su vez con las entidades del negocio; o sea las interfaces interactúan con las controladoras y con las entidades ya que los métodos de las controladoras lo que reciben como parámetros de entrada son objetos de tipo entidades. Las clases controladoras se relacionan con las entidades ya que reciben y devuelven objetos y listas de objetos de tipo entidades del negocio.

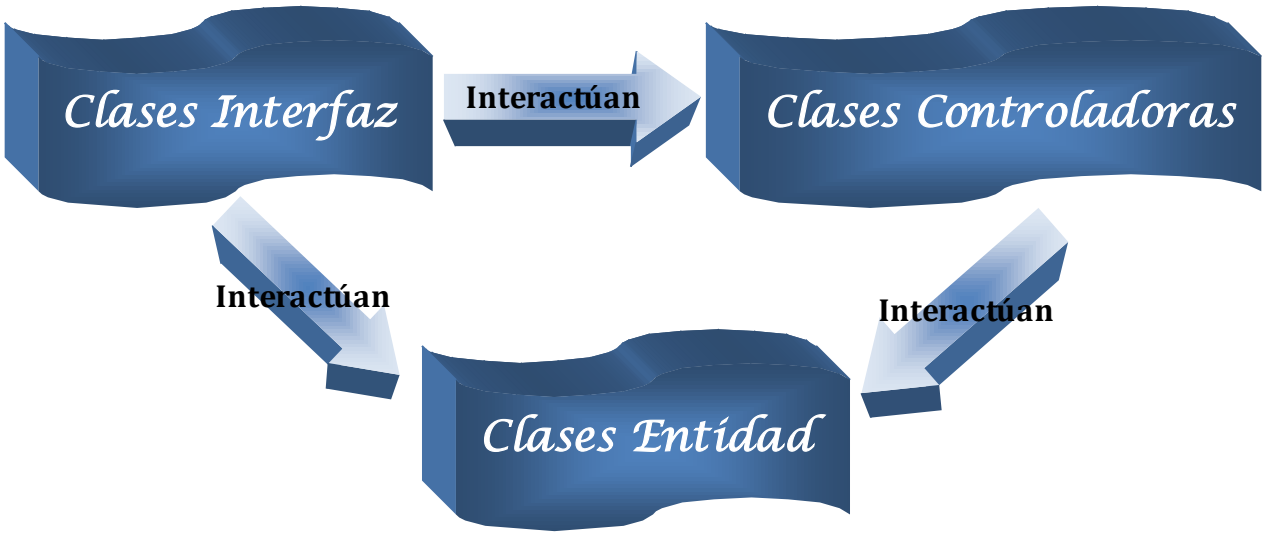

Fig. 10 Integración de clases en la Capa de Presentación

La capa de negocio se encuentra compuesta por las clases controladoras, las clases de acceso a datos y las entidades del negocio. Las clases controladoras interactúan con las clases de acceso a datos y ambas con las entidades del negocio ya que ellas tienen como parámetros de entrada y salida objetos o listas de objetos de tipo las entidades del negocio.

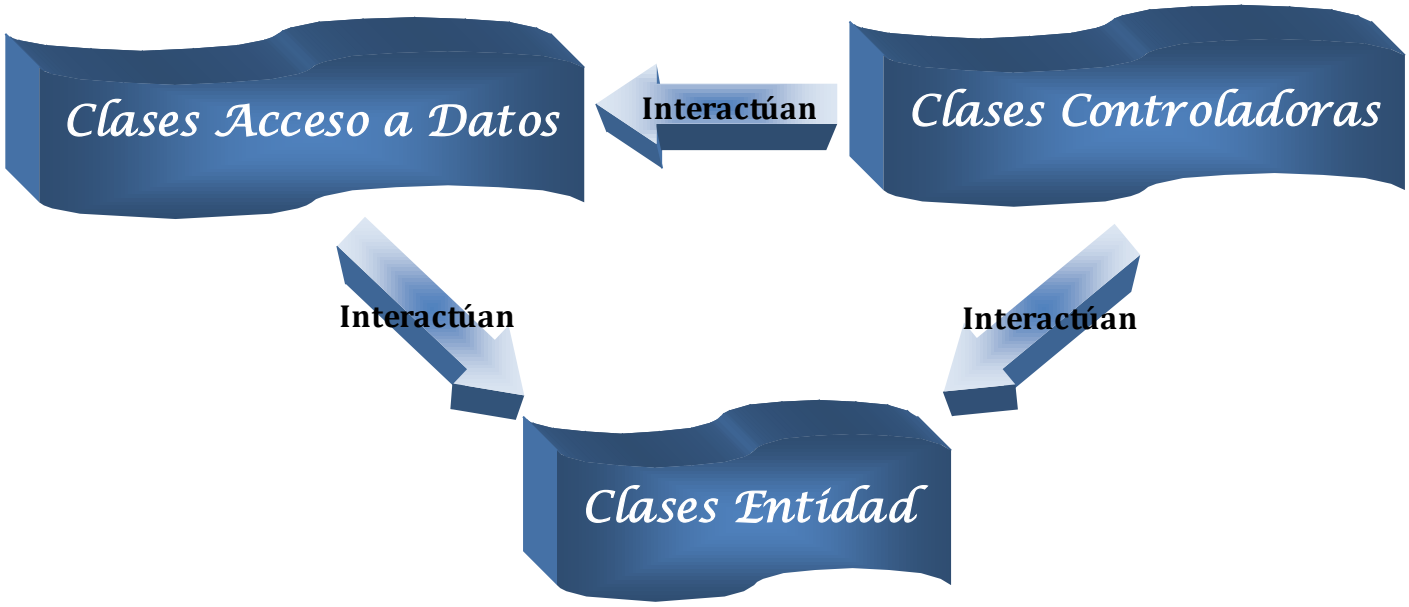

Fig. 11 Integración de clases en la Capa de Negocio.

Con la realización de este capítulo, se arribó a la conclusión, que la obtención del diseño propuesto por el analista resultó muy importante, pues permitió adquirir una comprensión de todo lo relacionado con los requisitos no funcionales y con las restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.

Además, al observar estilos de código se hace más legible el programa fuente, lo cual ayuda en la comunicación entre desarrolladores, y permite una temprana detección de errores. La propuesta ofrece estándares de codificación, por los que se rigen los programadores del proyecto, son completamente extensibles. Queda de parte de quienes la extiendan tratar de mantener un equilibrio, permitiendo una codificación homogénea y evitar las redundancias.

También resultó importante la descripción de los algoritmos no triviales tratando de lograr que se entienda como fue pensado y desarrollado el mismo. De la misma forma quedaron descritas las estructuras de datos que serán necesarias para la implementación del módulo, y las clases más importantes definidas en el diseño; tales como las clases entidad, interfaz y controladoras.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En el desarrollo del software, las posibilidades de error son innumerables. Pueden darse por una mala especificación de los requisitos funcionales, uso indebido de las estructuras de datos, errores al enlazar módulos, etc. El desarrollo del software ha de ir acompañado de alguna actividad que garantice la calidad; la prueba es un elemento crítico para la garantía de calidad del software.

La prueba y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo. Es fundamental medir la cobertura de las pruebas, es decir, la determinación de cuando se han realizado las suficientes pruebas. Si se siguen encontrando errores cada vez que se procesa el programa, las pruebas deben continuar.

Durante el mantenimiento debe de existir documentación de pruebas que incluya casos de prueba y resultados esperados. Si se producen modificaciones en el programa, habrá que probar de nuevo todas las partes del programa afectadas por las modificaciones.

Por lo dicho anteriormente, se desarrolló este capítulo, con el objetivo de especificar las pruebas que serán hechas al software y describir todo lo encontrado en las mismas.

Para ello fue necesario el estudio de varias herramientas que son usadas para la realización de pruebas de caja blanca mas conocidas como las Técnicas de cajas blancas o estructurales, y que se basan en un minucioso examen de los detalles procedimentales del código a evaluar. [3.2]

3.1 Pruebas

Pruebas de software es el conjunto de técnicas que permiten determinar la calidad de un producto software. Las Pruebas de software se integran dentro de las diferentes fases del Ciclo del software dentro de la Ingeniería de software. Así se ejecuta un programa y mediante técnicas experimentales se trata de descubrir que errores tiene.

La calidad de un sistema software es algo subjetivo que depende del contexto y del objeto que se pretenda conseguir. Para determinar dicho nivel de calidad se deben efectuar unas medidas o pruebas que permitan comprobar el grado de cumplimiento respecto de las especificaciones iniciales del sistema.(*Pruebas1*)

3.1.1 Objetivos

El objetivo de las pruebas de un programa es el de detectar todo posible malfuncionamiento antes de que entre en producción. Un error detectado en el laboratorio puede ser costoso de reparar; pero siempre es peor que el error le aparezca al usuario final.

En esta idea, una batería de pruebas será de mayor calidad cuantos menos errores queden por descubrir tras haberla pasado. Y, viceversa, si un programa aún tiene muchos fallos tras haber superado una batería de pruebas, se dirá que esta batería es de poca calidad.
Si se pudiera probar un programa con todos los posibles datos de entrada, se tendría una batería de pruebas perfecta, pues no hay lugar para las sorpresas. Lamentablemente, casi nunca es posible probar con todos los casos. En consecuencia se necesita un criterio para elegir qué casos se prueba. (*Información Objetivos de Pruebas*)

3.1.2 Alcance

Se lleva a cabo la prueba y se evalúan los resultados obtenidos frente a los resultados esperados. Si se descubren datos erróneos implica que hay un error y hay que corregirlo y empieza el proceso de depuración de errores.

Se basa en las estructuras de control del diseño procedimental para generar los casos de prueba que:

- Garanticen que se recorren por lo menos una vez todos los caminos independientes de cada módulo.
- Se ejecutan todas las decisiones lógicas en su parte verdadera y en su parte falsa.
- Se recorren todos los bucles.
- Se utilizan las estructuras de datos internas para garantizar su validez.
- Se invierte tiempo y esfuerzo en los detalles de control debido a que:
- Los errores suelen estar en situaciones fuera de las normales.
- A menudo caminos que se piensa que tienen pocas posibilidades de recorrerse, son recorridos regularmente.
- Los errores tipográficos son aleatorios. Puede que no sean detectados por los procesadores de la sintaxis del lenguaje particular y estar presentes en cualquier camino lógico.

3.2 Tipos de prueba.

- Pruebas de caja blanca
- Pruebas de caja negra

3.2.1 Pruebas de Caja Blanca

Si según se pasan las pruebas de caja negra, se tiene a mano un listado del programa y se van marcando las líneas de código que se ejecutan, se podrá determinar qué porcentaje de líneas ha sido ejecutado alguna vez, porcentaje que se conoce como cobertura de sentencias.

Puede que habiendo logrado una cobertura de caja negra del 100%, se mida una cobertura de sentencias del 100%. En este caso, se pasará a la fase siguiente: prueba de condiciones.

Pero puede ser que partes del código no hayan sido ejecutadas jamás (la cobertura de sentencias es inferior al 100%). En estos casos hay que pasar más pruebas buscando que se ejecuten todas y cada una de las sentencias del programa.

En el caso extremo de que no haya forma de ejecutar alguna parte del programa, se preguntarían si esa parte sirve para algo, o si se puede prescindir de ella.

Es muy recomendable alcanzar una elevada cobertura de sentencias, aunque no siempre es posible por premura de tiempo o medios.

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca se puede obtener casos de prueba que:

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

3.2.1.1 Prueba del camino básico.

La prueba del camino básico es una técnica de prueba de Caja Blanca.

Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Descripción de las pruebas de caja blanca realizadas.

Se llevó a cabo la prueba del camino básico que fue definido en el Epígrafe anterior al código del algoritmo FichaSinExamenLaboratorio.

.

Para aplicar la técnica del camino básico se debe introducir una sencilla notación para la representación del flujo de control, el cual puede representarse por un Grafo de Flujo.

Cada nodo del grafo corresponde a una o más sentencias de código fuente. Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo.

Prueba # 1

```
public List<FichaDonación> FichaSinExamenLaboratorio(FichaDonación fichaDonación)
      {
         List<FichaDonacion> fichaSinExamen = new List<FichaDonacion>(); 1
         PruebaLaboratorio pruebaLaboratorio = new PruebaLaboratorio();1

for (int i = 0; i < fichaDonacionRep.ObtenerTodos(fichaDonacion).Count; i++) 2
{
   pruebaLaboratorio.NroFicha=fichaDonacionRep.ObtenerTodos(fichaDonacion)[i].NroFicha;3
   if (pruebaLaboratorioRep.ObtenerTodos(pruebaLaboratorio).Count==0) 4
   {
      fichaSinExamen.Add(fichaDonacionRep.ObtenerTodos(fichaDonacion)[i]); 5
```

```
        }              6
    } 7
    return fichaSinExamen; 8
} 9
```

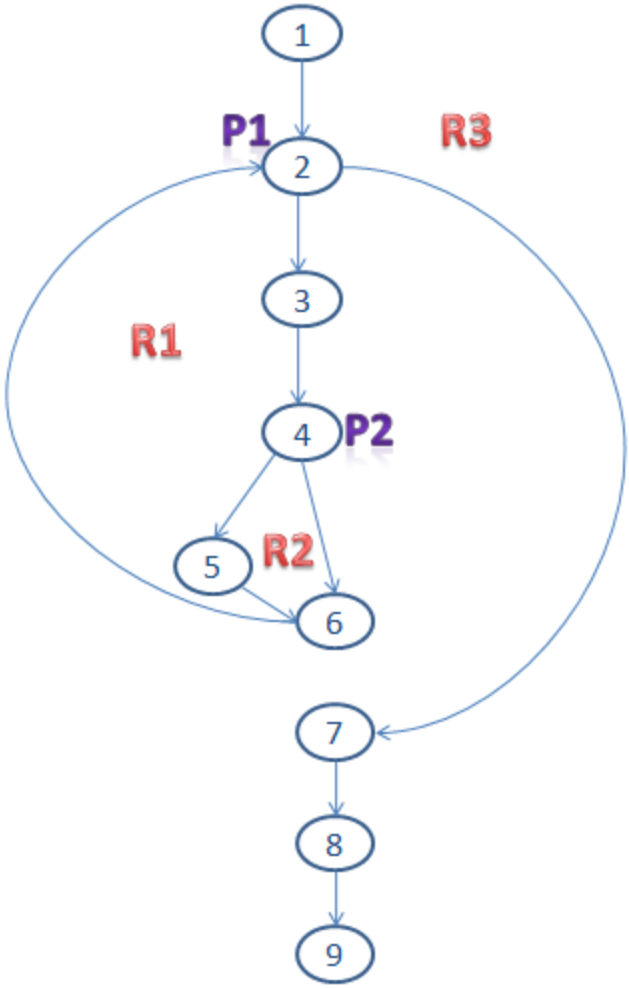

Fig. 12 Grafo de Flujo del Algoritmo FichaSinExamenLaboratorio

Luego de tener los grafos de Flujo se pasa al cálculo de la complejidad ciclomática de donde saldrán los caminos básicos a recorrer, las formulas para la obtención de complejidad son tres que deben llegar a un mismo numero de caminos.

1- V (G) = (A – N) + 2

Siendo "A" la cantidad total de aristas y "N" la cantidad total de nodos.

2- V (G) = P + 1

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

3- V (G) = R

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del calculo.

Calculo por la vía 1:

V (G) = (A – N) + 2

V (G) = (10– 9) + 2

V (G) = 3

Calculo por la vía 2:

V (G) = P+ 1

V (G) = 2 + 1

V (G) = 3

Calculo por la vía 3:

V (G) = R

V (G) = 3

Se obtuvieron tres caminos básicos para este código:

Camino 1: 1 2 3 4 5 6 2 7 8 9

Camino 2: 1 2 3 4 6 2 7 8 9

Camino 3: 1 2 7 8 9

Luego de tener elaborados los Grafos de Flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino.

Esto se lleva a cabo por los implementadores del sistema a través del tracio del código, sobre el software de desarrollo que se está usando: Visual Studio 2005.

Casos de prueba para el código del algoritmo FichaSinExamenLaboratorio

Caso de prueba realizado al Camino Básico 1.

Descripción: Los datos de entrada deben cumplir los siguientes requisitos: el usuario pasa una fecha de una solicitud, puede pasar el número de la ficha que es un entero mayor o igual a 1, o simplemente puede dejar los campos vacios.

Condición de ejecución: Debe haber al menos una ficha de donación en el sistema para que este método pueda obtener algún resultado.

Entrada: Se le entró una fecha.

Resultados Esperados: Se espera que muestre una lista con las fichas de donación para esta fecha que no tengan los resultados de los exámenes de laboratorio.

Con la ejecución del caso de prueba se obtuvieron todas las fichas que fueron realizadas en esa fecha, luego entra a un ciclo desde uno hasta la cantidad de fichas que se obtuvieron, una por una se busca en la tabla donde están los resultados de laboratorio y si esa ficha aun no tiene los resultados de laboratorio se pone en una lista que es la que se va a devolver. Como lo que se obtuvo fue el resultado esperado se califica la prueba como satisfactoria.

Caso de prueba realizado al Camino Básico 2.

Descripción: Los datos de entrada deben cumplir los siguientes requisitos: el usuario pasa una fecha de una solicitud, puede pasar el número de la ficha que es un entero mayor o igual a 1, o simplemente puede dejar los campos vacios.

Condición de ejecución: Debe haber al menos una ficha de donación en el sistema para que este método pueda obtener algún resultado.

Entrada: Se le entró una fecha

Resultados Esperados: Se espera que todas las fichas ya tengan incluidos los resultados de los exámenes hechos y por tanto no me retorne nada.
Con la ejecución de este caso de prueba se obtuvieron todas las fichas que fueron realizadas en esa fecha, luego al entrar al ciclo desde uno hasta la cantidad de fichas que se obtuvieron, una por una se busca en la tabla donde están los resultados de laboratorio y si existe alguna ficha que no tenga los resultados de los exámenes por este camino no los obtendré porque el mismo no incluye el adicionar ficha sin resultados de exámenes.

Caso de prueba realizado al Camino Básico 3.

Descripción: Los datos de entrada deben cumplir los siguientes requisitos: el usuario pasa una fecha de una solicitud, puede pasar el número de la ficha que es un entero mayor o igual a 1, o simplemente puede dejar los campos vacios.

Condición de ejecución: Debe haber al menos una ficha de donación en el sistema para que este método pueda obtener algún resultado.

Entrada: Le entro el número de una ficha.

Resultados Esperados: Se espera que me devuelva una ficha para verificar si tiene los resultados de los exámenes hechos.

Con la ejecución de este caso de prueba se obtuvo la ficha de donación pero no se pudo verificar por este camino básico si tenia los resultados de las pruebas, por esto se cree que este camino es innecesario para la obtención de resultados de este algoritmo.

3.2.2 Pruebas de Caja Negra

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del *software*. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.
La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca.

Muchos autores consideran que estas pruebas permiten encontrar: [Pressman, 1998] [MYE, 1979][Beizer,1995]

- ✓ 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.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están: [Pressman, 2000]

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

Descripción de la prueba de caja negra realizada.

Para la aplicación de este tipo de prueba se usará el caso de uso “Registrar Solicitud”, este caso de uso está basado en las gestiones necesarias para realizar un pedido de productos por parte de un servicio hospitalario, que puede ser una sala, un laboratorio o un salón de operaciones, así como los procesos que realiza el farmacéutico para despachar estos productos.

Clases Validas	Clases no Validas	Resultados	Evaluación	Observaciones
El usuario da clic en el botón “Datos del Paciente” y se le muestra la ventana que le permite la	El usuario al dar clic en el botón datos del paciente no se le muestre la ventana para la entrada de	Luego de dar clic en el botón el usuario ve la ventana para el llenado de los datos.	Satisfactorio	La operación se realizó correctamente

entrada de los datos del paciente.	datos del paciente.			
El usuario da clic en el botón “Componentes sanguíneos a solicitar” y se le muestra la ventana que le permite elegir, él o los componentes que solicitará así como sus respectivas cantidades.	Al dar clic en el botón Componentes sanguíneos a solicitar no se muestren los componentes sanguíneos.	El usuario pudo seleccionar los componentes sanguíneos	satisfactorio	La operación se realizó correctamente.
El usuario da clic en el botón “Enviar” creando con todos los datos escritos y seleccionados una nueva solicitud en la base de datos.	Al dar clic en el botón no se crea la solicitud.	Al dar clic en el botón “Enviar” se crea la nueva solicitud en la base de datos	satisfactorio	Al insertar los datos del paciente y la solicitud y al escoger los componentes que se necesitaran se pudo probar que quedo creada una nueva solicitud en la base de datos.
El usuario introduce el nombre de un paciente para el que ya se ha hecho una solicitud	No muestra los datos de la solicitud hecha por este paciente para ser modificada.	Se muestran los datos de la solicitud que pueden ser cambiados y se actualiza la solicitud.	satisfactorio	

El usuario decide cancelar el trabajo que iba a realizar al dar clic en el botón “Cancelar”	Se insertan los datos que habían sido seleccionados o escritos en las ventanas.	Queda cancelado lo que se había escrito o marcado en las ventanas de esta pagina y no se crea la solicitud.	Satisfactorio.	

Con la realización de este capítulo, se ha podido arribar a la conclusión que la elaboración de las pruebas resulta de gran importancia para la validación de la calidad del software. Es por ello, que se desarrollaron dos tipos de prueba a la aplicación, donde se obtuvieron resultados satisfactorios, pues no se encontraron errores de implementación lo que muestra que el trabajo de desarrollo se llevo a cabo con el cuidado requerido.

CONCLUSIONES

Los sistemas que se encuentran desarrollados para dar solución a la problemática existente en los Bancos de Sangre a nivel mundial no cumplen, por una razón u otra, con las expectativas que hoy se plantean. Es por esto, que ha resultado de gran importancia la implementación del sistema informático para la gestión de los procesos de información hospitalaria en los Bancos de Sangre de los hospitales se desarrolló en la presente Tesis.

Para lograr que este resultado llegara a ser el más óptimo, se cumplieron las tareas previstas:

- Se realizó un análisis satisfactorio de las tecnologías y herramientas a usar.
- Se implementó usando el patrón de diseño definido por el analista.
- Se desarrolló una aplicación que permite la gestión de la información.
- Se implementó además, una interfaz orientada al usuario para fácil manejo del sistema.
- Se garantizó la seguridad de los datos.
- Se realizaron pruebas que resultaron necesarias para garantizar el correcto funcionamiento de la aplicación implementada.

Al dar cumplimiento a las tareas antes mencionadas se obtuvieron los resultados siguientes:

- Una aplicación Web que agiliza el trabajo del personal que se encuentra vinculado al Departamento del Banco de Sangre.
- El logro de que los datos que son archivados en la base de datos a través de la aplicación, se puedan obtener en las próximas donaciones y su obtención resulte fácil.
- Una solución óptima a las asignaciones por solicitudes que realiza este departamento por parte de cualquier departamento del hospital. Esto a su vez pone de manifiesto la integración del sistema con otros sistemas que conforman el sistema de gestión hospitalaria en general.
- La existencia de otras funcionalidades que no se desarrollaron conllevaron a que se alcanzara un 85% de la implementación, dándole solución a gran parte de los requerimientos funcionales que fueron planteados.

RECOMENDACIONES

Se recomienda luego de la realización del trabajo:

A la Dirección de la Facultad:

- Continuar promoviendo la vinculación temprana, por parte de los estudiantes de la Universidad y en especial de la facultad, a proyectos productivos que ayuden en la informatización de la salud.
- Continuar el desarrollo del sistema que ha sido realizado con vistas a lograr la totalidad del cumplimiento de los requisitos funcionales que fueron planteados.

A la Dirección del Proyecto:

- Dar continuidad al desarrollo del algoritmo de "Asignaciones por solicitudes" hasta obtener la solución más óptima, pues resulta un proceso de gran importancia entre los que se efectúan en el Banco de Sangre.
- Proporcionar asesoramiento técnico especializado en la materia para lograr un mayor entendimiento de los procesos que se llevan a cabo en el Banco de Sangre.

BIBLIOGRAFÍA

Archer, T. (2001).A Fondo C#

Arora, G., B.Alaswany, et al. (2002).C#

ADO. Disponible en: http://es.wikipedia.org/wiki/ADO

Ajax. Disponible en: http://es.wikipedia.org/wiki/AJAX

Byte_Codes. Disponible en: http://es.wikipedia.org/wiki/Byte_codes

Concepto Banco de Sangre. Disponible en:
http://www.cardiologia.org.mx/incic/ban_sangre/tex_sangre.htm

ECMA. Disponible en: http://es.wikipedia.org/wiki/ECMA

Eiffel. Disponible en: http://es.wikipedia.org/wiki/Eiffel

Estructura de Datos. Disponible en: http://es.wikipedia.org/wiki/Estructura_de_datos

Fergunson, J., B. Patterson, et al (2003). La Biblia de C#

FreeBSD. Disponible en: http://es.wikipedia.org/wiki/FreeBSD

Glóbulos Blancos. Disponible en: http://es.wikipedia.org/wiki/Gl%C3%B3bulos_blancos

Glóbulos rojos. Disponible en: http://es.wikipedia.org/wiki/Gl%C3%B3bulos_rojos

Grupo Sanguíneo. Disponible en: http://es.wikipedia.org/wiki/Grupo_sangu%C3%ADneo

Información de CSharp. Disponible en: http://www.desarrolloweb.com/articulos/561.php

Informacion de Mono. Disponible en:
http://www.mentores.net/articulos/Mono_Net_Framework_plataformas_noWindows.htm

Informacion Objetivos de Pruebas. Disponible en:
http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/calidad.htm

Información Tipos de Prueba. Disponible en:
http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/calidad.htm

Java. Disponible en: http://es.wikipedia.org/wiki/java

LINUX. Disponible en: http://es.wikipedia.org/wiki/linux

MacOSX. Disponible en: http://es.wikipedia.org/wiki/MacOSX

Mono Basic. Disponible en: http://es.wikipedia.org/wiki/MonoBasic

.Net. Disponible en: http://es.wikipedia.org/wiki/.net

OASIS. Disponible en: http://es.wikipedia.org/wiki/OASIS

Ojeda, F. C. (2002). Visual C#. Net

Plaquetas. Disponible en: http://es.wikipedia.org/wiki/Plaquetas

Pruebas1. Disponible en: http://es.wikipedia.org/wiki/Pruebas_de_Software

Python. Disponible en: http://es.wikipedia.org/wiki/Python

Reutilización. Disponible en: http://www.es.gnome.org/documentacion/articulos/bonobo-why/bonobo-why/x20.html

Sangre. Disponible en: http://es.wikipedia.org/wiki/Sangre

Solaris. Disponible en: http://es.wikipedia.org/wiki/Solaris

Transfusión de sangre. Disponible en: http://es.wikipedia.org/wiki/Transfusi%C3%B3n_de_sangre

W3C. Disponible en: http://es.wikipedia.org/wiki/W3C

WS SECURITY. Disponible en: http://es.wikipedia.org/wiki/WS_Security

GLOSARIO DE TÉRMINOS

ADO (ActiveX Data Objects) es uno de los mecanismos que usan los programas de computadoras para comunicarse con las bases de datos, darles órdenes y obtener resultados de ellas. Con ADO, un programa puede leer, insertar, editar, o borrar, la información contenida en diferentes áreas de almacenamiento dentro de la base de datos llamadas tablas. Además, se puede manipular la propia base de datos para crear nuevas áreas para el almacenamiento de información (tablas), como también alterar o eliminar las ya existentes, entre otras cosas. (*ADO*)

Byte-codes, que son el resultado de la compilación de un programa Java. (*Byte_Codes*)

Ecma International es una organización internacional basada en membresías de estándares para la comunicación y la información. Adquirió el nombre *Ecma International* en 1994, cuando la *European Computer Manufacturers Association* (ECMA), cambió su nombre para expresar su alcance internacional. Como consecuencia de esta decisión, el nombre ya no se considera un acrónimo y no se escribe completamente en mayúsculas. (*ECMA*)

Eiffel es un lenguaje de programación orientado a objetos centrado en la construcción de software robusto. Su sintaxis es parecida a la del lenguaje de programación Pascal. Una característica que lo distingue del resto de los lenguajes es que permite el diseño por contrato desde la base, con precondiciones, postcondiciones, invariantes y variantes de bucle, invariantes de clase y asertos, es un lenguaje con tipos fuertes, pero relajado por herencia. Implementa administración automática de memoria, generalmente mediante algoritmos de recolección de basura. Las claves de este lenguaje están recogidas en el libro de Meyer, *Construcción de Software Orientado a Objetos*. (*Eiffel*)

FreeBSD es un sistema operativo libre para ordenadores personales basado en las CPU de arquitectura Intel, incluyendo procesadores 386, 486 (versiones SX y DX), y Pentium. También son soportados los procesadores compatibles con Intel como AMD y Cyrix. Actualmente también es posible utilizarlo hasta en 10 arquitecturas distintas como Alpha, AMD64, IA-64, MIPS, PowerPC y UltraSPARC. (*FreeBSD*)

GNU/Linux (*GNU con Linux* o *GNU+Linux*) es la denominación defendida por Richard Stallman y otros para el sistema operativo que utiliza el kernel Linux en conjunto con las aplicaciones de sistema creadas por el proyecto GNU y de varios otros proyectos/grupos de software. Comúnmente este sistema operativo es denominado como Linux, aunque Stallman sostiene que esta denominación no es correcta. (*LINUX*)

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los 90. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es interpretado (usando normalmente un compilador JIT), por una máquina virtual Java. (*Java*)

MonoBasic la versión para mono de Visual Basic. (*Mono Basic*)

Mac OS X es el actual sistema operativo de la familia de ordenadores Macintosh(MacosX).

.NET Es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de aplicaciones. Basado en esta plataforma, Microsoft intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el Sistema Operativo hasta las herramientas de mercado. (*.Net*)

OASIS, acrónimo de (Organization for the Advancement of Structured Information Standards) es un consorcio internacional sin fines de lucro que orienta el desarrollo, la convergencia y la adopción de los estándares e-business.OASIS permite realizar eficazmente aplicaciones JAVA. (*OASIS*)

Python es un lenguaje de programación creado por Guido van Rossum en el año 1990[1]. Es comparado habitualmente con TCL, Perl, Scheme, Java y Ruby. En la actualidad Python se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation. (*Python*)

Solaris es un sistema operativo desarrollado por Sun Microsystems. Es un sistema certificado como una versión de UNIX. Aunque Solaris en sí mismo aún es software propietario, la parte principal del sistema operativo se ha liberado como un proyecto de software libre denominado *Opensolaris*. Solaris puede considerarse uno de los sistemas operativos más avanzados. Sun denomina así a su sistema operativo(*Solaris*).

Transfusión de sangre. La transfusión de sangre es el procedimiento médico de incorporar sangre o sus derivados procedentes de un individuo en el sistema circulatorio de otro, utilizado para mantener con vida a los pacientes que han sufrido pérdidas excesivas de sangre por traumas o cirugía, o para proporcionar algún elemento necesario en caso de enfermedades que afectan la producción. (*Transfusión de sangre*)

WS-Security (**Seguridad en Servicios Web**) es un protocolo de comunicaciones que suministra un medio para aplicar seguridad a los Servicios Web. En Abril de 2004 el estándar WS-Security 1.0 fue publicado por Oasis-Open. En 2006 fue publicada la versión 1.1. (*WS SECURITY*)

El **World Wide Web Consortium**, abreviado **W3C**, es un consorcio internacional que produce estándares para la World Wide Web. Está dirigida por Tim Berners-Lee, el creador original de URL (*Uniform Resource Locator*, Localizador Uniforme de Recursos), HTTP (*HyperText Transfer Protocol*, Protocolo de Transferencia de HiperTexto) y HTML (Lenguaje de Marcado de HiperTexto) que son las principales tecnologías sobre las que se basa la Web. (*W3C*)