UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 7
INGENIERÍA INFORMÁTICA

IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN DE LABORATORIOS CLÍNICOS (LIS)

TRABAJO DE DIPLOMA PARA OPTAR POR EL
TÍTULO DE INGENIERÍA EN CIENCIAS INFORMÁTICAS.

Autor: Alnair Reyes Pérez

Tutor: Ing. William Soñora

Ciudad de La Habana
Julio del 2007

“Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber”

Albert Einstein

Declaración de Autoría

Declaro ser autor de la presente tesis y reconozco 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 5 días del mes de Julio del año 2007.

_____	_____
Alnair Reyes Pérez	Ing. William Soñora Cruz

Dedicatoria

A mi Madre... solo existe una palabra para definirte, A M O R,

muchas gracias por todo mami.

A mi hermanita Anailys...porque es y siempre será mi nene suavecita,

te quiero mucho pupu.

A mi abuelita Juanita... por esos besitos tan ricos que me da,

por ser mi viejita, la más linda del mundo.

Agradecimientos

Resumir en algunos nombres de personas e instituciones todo el agradecimiento por el alcance de un sueño, es una tarea bastante difícil, quizás más que su realización. En este empeño por crecer han estado involucradas muchas personas que han sido el sustento y aliento diario a no cejar en el empeño. En primer término quiero mencionar a mi madre, a quien no solo agradezco sus consejos y el apoyo constante, sino además, y sobre todas las cosas, el privilegio de ser su hijo.

A mi hermana Anailys, mi abuela Juanita y mi papá José, a mi segundo papá Carlos, mis hermanitas Claudia y Nerlita, a mi hermanito Joseito, a mis tíos Icho y Tata, a mis primos queridos Yalien, Yaniarys, Yadianys, Jorlan y Jordan, a mis abuelos Verde Olivo y Pastor, a mi vecina Cuca, a Nerlys.

A mis amigos Víctor, Ariel y Yordanys, más que amigos, hermanos, fieles cómplices y compañeros incondicionales, que asumieron conmigo muchos sacrificios. A mis amigos y compañeros de grupo de estos 5 largos años, Ariesky, Eduardo, y Abel.

A mi tutor William, y a todos los integrantes del proyecto de hospitales de la facultad 7. A mis muchos profesores, artífices y guías en mi preparación.

A todos los amigos y amigas que compartieron trabajo, amor, sonrisas, consejos y momentos inolvidables.

Un agradecimiento muy especial a mi bisabuelita Mimí, que en paz descanse, la quise mucho, por el amor y alegría nos dio a todos sus bisnietos, sus nietos y su hija.

Y finalmente, quiero agradecer a la Revolución y a nuestro Comandante Fidel Castro por haber acercado tanto, tan solo a la distancia del empeño, los sueños de muchos jóvenes que, como yo, están convencidos que el futuro se construye con nuestras propias manos.

Muchas gracias por estar,
Alnair Reyes Pérez.

Resumen

Actualmente, en los laboratorios clínicos de los hospitales cubanos no existe un sistema informático, que gestione los procesos que se realizan en los mismos, el manejo de la información, ya sea de los análisis o del control de los equipos y reactivos se tiene que realizar de forma manual. Lo que trae consigo demoras y un mal manejo de dicha información. Para solucionar este problema, se plantea como objetivo del presente trabajo, implementar una aplicación Web, que presente las Funcionalidades que se requieren para brindar un mejor servicio en estos laboratorios.

Entre las principales herramientas utilizadas en el desarrollo de la aplicación se encuentra la plataforma .NET de Microsoft, usando C# como lenguaje de programación desarrollador del negocio en conjunto con la herramienta AJAX, así como JavaScript para las validaciones por la parte del cliente. Como Sistema Gestor de Base de Datos se uso PostgreeSQL, se implementó una arquitectura de 3 capas. Los artefactos de la ingeniería de software propuestos por el analista fueron modelados usando la metodología RUP.

El sistema esta siendo sometido a un proceso de pruebas, y en esta etapa mas específicamente las de unidades (caja blanca y caja negra), dejando para próximas iteraciones del desarrollo las de integridad y aceptación.

Se espera que el sistema sea usado en los centros hospitalarios del país, optimizando así los servicios que se brindan a los pacientes y perfeccionando los escritorios de trabajo de los profesionales de la salud.

Índice

Introducción

El Sistema Nacional de Salud de Cuba, es universal, gratuito y al alcance de todos los cubanos sin distinción de raza, procedencia social o religión. Se ha conformado y desarrollado a partir de un concepto social de la salud que en el momento actual, es preciso decir que, rebasa los límites del individuo y que abarca su relación e interacción con el medio donde éste se desarrolla.

Desde el mismo año 1959, con el triunfo de la Revolución, la salud pública cubana ha mostrado una clara vocación por orientarse hacia la comunidad. Desde la promulgación de la Ley del Servicio Médico Rural y la creación de los hospitales rurales en la década del 60 hasta nuestros días. [1]

El Ministerio de Salud Pública de Cuba, desde el año 2000 ha definido la informatización del sector de la salud, en busca de la optimización de los servicios que se brindan a la población, mayor productividad y competencia en el desempeño de sus profesionales y técnicos, control en la administración de sus recursos, eficacia.

Se han estado produciendo y poniendo en práctica soluciones informáticas para la automatización de algunos procesos administrativos, docentes, asistenciales e investigativos. El uso de la Informática en la Medicina es una de las aplicaciones más comunes e importantes desde hace varias décadas. Ha permitido al sector de la salud, no sólo contar con métodos novedosos, sencillos y eficaces de gestión administrativa en consultas, hospitales y centros de investigación biomédica; sino también disponer de complejos software que reducen la posibilidad de error en el diagnóstico de las enfermedades y que aceleran su formulación.

Para llevar a cabo el Proceso de Informatización del SNS abarcando al hospital como eje fundamental, fue encomendada esta tarea por la dirección del MINSAP y el Ministerio de Informática y Comunicaciones (MIC), a la Universidad de las Ciencias Informáticas(UCI) y mas especifico a la facultad 7, dedicada a la ejecución de soluciones informáticas para la salud, que en conjunto con Médicos Especialistas en Medicina General Integral en calidad de expertos funcionales, tienen la misión en el marco del Proyecto de Sistema de Gestión Hospitalaria (GeHos) de elaborar un producto de software que facilite la gestión de la información y la toma de decisiones en este nivel de atención.

INTRODUCCION

El laboratorio clínico es una especialidad de la Medicina que se ocupa de realizar el diagnóstico, el pronóstico y la evolución de las enfermedades mediante el uso de diferentes técnicas. Esta especialidad comienza a adquirir identidad propia como tal a principios del siglo XX.

Con el amplio desarrollo que ha alcanzado el sistema de salud cubano se ha logrado la creación de laboratorios clínicos distribuidos por todo el país, así como de numerosos centros de investigación los cuales cuentan con profesionales que además de una elevada preparación científica poseen una gran calidad humana. Estos laboratorios tienen como objetivo, fundamentalmente, la asistencia médica a la población.

A pesar que se han automatizado varios servicios en muchos laboratorios, aun falta mucho por lograr en este sentido, y de ahí se tiene la siguiente **situación problémica** en cuestión ya que en los laboratorios clínicos es muy difícil llevar un control de toda la información que hay en movimiento. Las secretarias que toman las muestras de los análisis tienen problemas porque los libros son muy grandes y no hay modelos oficiales en existencia para esta tarea. Dentro del laboratorio, los procedimientos para el trabajo con los análisis son muy engorrosos además hay que hacer una copia de los datos que toma la secretaria agregándole los resultados de los exámenes realizados es decir se esta haciendo el mismo trabajo doble.

Los laboratorios trabajan con diferentes reactivos y productos de los cuales es necesario tener un control estricto y esto se esta haciendo manualmente lo que dificulta el trabajo estadístico sobre la planificación y historiales de consumo por la perdida de tiempo y esfuerzo del personal implicado en la tarea. También, es necesario llevar una serie de estadísticas sobre los pacientes que se tratan, sus enfermedades, análisis y todos los procesos en general del laboratorio, estas estadísticas se dificultan a niveles extremos por la cantidad de pacientes y muestras de análisis con los que se trabajan. La coordinación en general, es difícil de concertar a no ser hablando directamente con cada persona.

Teniendo en cuenta lo anteriormente expuesto y la necesidad de buscar una respuesta a estas dificultades, que frenan el desarrollo eficiente de las actividades de los laboratorios clínicos, los esfuerzos estarán encaminados a resolver el siguiente **problema:** ¿Cómo automatizar el proceso de gestión de información de los laboratorios clínicos para lograr que el servicio sea más rápido y eficiente?

INTRODUCCION

De acuerdo con los estudios realizados, no se ha llevado a término ningún trabajo encaminado a solucionar el problema anteriormente mencionado.

Por lo que, como parte del proceso de informatización y a partir de la solución del problema planteado, se espera como **aporte práctico** el desarrollo de un sistema de información y manejo administrativo para los Laboratorios Clínicos, de fácil manejo y amigable para el usuario final, que cumpla con todas las necesidades de la vida diaria en dichos laboratorios y gestione eficientemente y de forma segura toda la información que se genera en el proceso de atención a la población, y que a la vez sea capaz de brindar dicha información a todos los niveles de toma de decisiones del SNS.

Como **objeto de estudio** se ha identificado los procesos de gestión de la información vinculados al hospital y la especificación de las herramientas informáticas a utilizar para el desarrollo del módulo.

Como **campo de acción**, el proceso gestión de la información correspondiente a los laboratorios clínicos.

Se propone como **Objetivo General:** Desarrollar un sistema informático, para la gestión y procesamiento de la información del laboratorio.

Teniendo en cuenta el objetivo general expuesto anteriormente se plantean los siguientes **Objetivos Específicos:**

- Realizar un estudio detallado del proceso de gestión de la información en los laboratorios clínicos.
- Estudiar el diseño y arquitectura definidos para el sistema.
- Implementar el sistema según el diseño y la arquitectura definidos.
- Analizar críticamente la tecnología utilizada.
- Realizar un análisis de la integración con otros componentes o partes del sistema.
- Analizar los algoritmos usados, ya sean conocidos o nuevos, y demostrar su eficiencia.
- Utilizar los patrones de diseño.
- Realizar las pruebas al sistema definiendo la estrategia, diseño y ejecución de las mismas y realizar un análisis de los resultados.
- Cumplir con los estándares definidos por la Facultad 7 y la dirección del proyecto.
- Programar una interfaz vistosa, clara y amigable que satisfaga las necesidades del cliente.

Para lograr los objetivos anteriormente planteados se trazaron las siguientes **tareas**:

- Hacer un estudio del problema de los laboratorios y la situación actual.
- Realizar un estudio profundo sobre los aspectos con los que debe contar el sistema, los requisitos que debe cumplir.
- Realizar un estudio previo de los laboratorios clínicos y como informatizarlos Definir tecnología a utilizar; ver sistemas previos y antecedentes en esta esfera.
- El análisis de sistemas previos de los laboratorios arrojó que en el mundo hay varios sistemas similares pero sobre el sistema cubano que ya esta desarrollado (GalenLab) hay que hacer un estudio particularmente profundo.
- Realizar un estudio minucioso de la bibliografía relacionada con los temas de las herramientas, tendencias y tecnologías actuales.
- Realizar un estudio de los flujos de trabajo que se recogen en el proceso de desarrollo RUP y del lenguaje de modelado UML.
- Diseñar y elaborar un prototipo para presentar al usuario y determinar el cumplimiento de las funcionalidades estudiadas.
- Desarrollar un sistema con calidad, portable, reutilizable con seguridad y una interfaz amigable.

Para realizar una descripción detallada, este documento muestra el resultado de la investigación de la siguiente forma:

CAPÍTULO 1: Contiene la fundamentación teórica del tema tratado en la investigación. Se hace una descripción de las tendencias y tecnologías actuales que se utilizaron como soporte de la propuesta.

CAPÍTULO 2: Se realiza un análisis crítico de los artefactos de la ingeniería de software que fueron entregados por el analista. Se hace una descripción de los algoritmos no triviales implementados, así como un análisis de las principales estructuras de datos usadas y finalmente se hace una descripción de las principales clases que se utilizaron.

CAPITULO 3: Se hace un análisis de los casos de prueba que se le aplicaron al sistema y la presentación de algunos resultados obtenidos.

El documento cuenta además con las conclusiones del trabajo, algunas recomendaciones a tener en cuenta en la continuidad del Proyecto y un conjunto de anexos que permitirán una mejor ilustración del estudio realizado.

CAPITULO 1

Fundamentación teórica del tema

Introducción

En este capítulo se aborda el estado actual de desarrollo de las aplicaciones de gestión de laboratorios clínicos en el ámbito mundial, y nacional, además de un estudio de las tecnologías en las que se apoya el desarrollo del sistema en función de un análisis de las tendencias actuales.

Han sido implementados varios sistemas de gestión de laboratorios clínicos, que de una forma u otra responden a las exigencias, cada día más crecientes, de responsabilidad, calidad y rapidez con los servicios de la salud; de estructuras muy variadas donde cada uno le da el enfoque de la actividad que necesita informatizar, se han aplicado fundamentalmente las tecnologías Web.

1.1 Estado del Arte

La importancia que han adquirido en la sociedad la informatización de los servicios de la salud y fundamentalmente los laboratorios clínicos, ha provocado que en el mundo hayan surgido un sin número de sistemas automatizados dirigidos al control eficiente y veloz de dichos procesos.

A continuación, se hace un estudio crítico sobre los principales sistemas existentes en el ámbito nacional e internacional para la gestión de la información en los laboratorios clínicos.

Ámbito Internacional

TIMLAB Solución para laboratorios clínicos.

Desarrollado por TIMSA Software, es una solución que permite manejar fácilmente la administración de las operaciones habituales de un laboratorio. El objetivo principal es garantizar seguridad e imagen en los resultados clínicos y administrativos. [2]

Las versiones disponibles de TIMLAB son desarrolladas bajo el asesoramiento de expertos en medicina para garantizar su funcionalidad. La nueva versión TIMLAB Web, adiciona a los beneficios que ofrecen las otras versiones, permite tanto a los médicos, pacientes, laboratorios de maquila y empresas consultar los resultados a través de Internet.

TIMLAB esta desarrollado con la siguiente tecnología:

Lenguaje Orientado a Objetos: Java J2EE en modelo MVC.

Soporte para Bases de Datos: Progress, DB2, Oracle, MySQL, PostgreSQL.

Sistemas operativos: Windows, UNIX, Linux.

Versiones disponibles:
- Cliente- Servidor.
- Web. (Disponible en varios idiomas)
- Hand held.

Care2x

Care2x fue concebido y desarrollado en el año 2002, por Elpidio Latorilla y cuenta con un grupo de desarrollo de más de 100 integrantes procedentes de más de 20 países.

Care2x es un sistema informático integrado. Care2x integra datos, funciones y flujo de tareas en un entorno de cuidados de la salud. Al momento, está compuesto de cuatro componentes principales. Cada uno de estos componentes también puede funcionar de manera individual.[3]

- HIS - Sistema de Información Hospitalaria/Servicios de la Salud
- PM - Administración del Ejercicio Médico
- CDS - Servidor Central de Datos
- HXP - Protocolo de Intercambio de Datos de la Salud

Plataforma de desarrollo: PHP 4

Soporte para bases de datos: MySQL, PostgreeSQL,

Servidor Web: Apache 1.x o 2.x.

Se encuentra bajo los estándares y beneficios del software libre, derivado del open source y con toda la potencia de GNU/Linux. Es entregado en una distribución especialmente desarrollada por BioLinux.

CONLAB LIS® - Laboratorio

Es un sistema de información y manejo administrativo para Laboratorios Clínicos, de fácil manejo y muy amigable para el usuario final debido a su sistema autodidáctico. Esta escrito con productos de Microsoft®, lo cual garantiza un constante mejoramiento de la aplicación a costos razonables. Cumple con todas las necesidades de la vida diaria de los Laboratorios Clínicos de Centro, Sur América y Europa, pues ha sido diseñado luego de un estudio de las necesidades de la región. [4]

Plataforma de desarrollo: Visual Basic.

Soporte para bases de datos: Access, Sql Server

Sistema operativo: Windows NT/2000.

Ámbito Nacional

GalenLab

Desarrollado por Softel, está diseñado para ser utilizado por los técnicos, médicos, enfermeras y personal administrativo de Medios de diagnósticos independientes o pertenecientes a un centro hospitalario, que necesitan del sistema para optimizar su trabajo y elevar su eficiencia.

Cuenta con un sistema de ayuda en línea que brinda al usuario toda la información que necesite sobre el proceso que esta efectuando y posee un estricto control de acceso que permite a cada técnico visualizar solamente la información y opciones del sistema relacionadas con su actividad. No permite el acceso a ninguno de los módulos e informaciones a partir de puntos no autorizados.

Plataforma de desarrollo: Visual Basic 6.0.

Soporte para bases de datos: SQL Server 2000.

Sistema operativo: Windows (NT o superior, 98 o superior).

1.2 Herramientas utilizadas para el desarrollo del sistema.

Entre las herramientas que actualmente se utilizan en el mundo por los desarrolladores, se encuentran fundamentalmente:

- SQL Server, MySQL, Oracle y PostgreeSQL como gestores de bases de datos.
- Apache e IIS (Internet Information Service) como servidores Web.
- Como plataformas de desarrollo se usan PHP 4, J2EE, y Visual Basic básicamente.

Y se ha incorporado desde hace muy poco tiempo una muy potente herramienta: Visual Studio .NET que compite con el J2EE (Java Enterprise Edition).

Para el desarrollo del sistema se realizó un estudio sobre las posibles herramientas a utilizar en su construcción. Teniendo en cuenta las tendencias actuales y las novedades en este campo.

1.2.1 La tecnología .NET

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años, con el objetivo de obtener una plataforma sencilla y potente para distribuir el software en forma de servicios, de forma tal que puedan ser suministrados remotamente, comunicándose y combinándose unos con otros totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados.

.NET ofrece un entorno de desarrollo de aplicaciones llamado *Visual Studio* .NET que consta de varios lenguajes de programación como Visual Basic .NET, Visual C#, Visual FoxPro y Visual C++ .NET. Estos lenguajes combinan las características de los lenguajes existentes con nuevas posibilidades para

proporcionar un potente sistema de desarrollo. A continuación, se detallan algunas de las características de la Arquitectura .NET.[5]

1.2.1.1 Arquitectura Framework.NET

La arquitectura .NET (.NET Framework) es el modelo de programación de la plataforma .NET para construir y ejecutar los servicios .NET. El objetivo de esta arquitectura es la de reducir la complejidad en el desarrollo de este tipo de aplicaciones, permitiendo a los desarrolladores centrarse en escribir la lógica específica del servicio a desarrollar.

1.2.2 Lenguajes de programación

Lenguaje C#

Microsoft ha creado C# que combina algunas de las características más avanzadas de Java con algunas de las más potentes de C y C++, construido especialmente para adaptarse de manera natural al framework.

Las principales características que identifican al lenguaje C# son:

Sencillez: Elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET.

Modernidad: Incorpora en el propio lenguaje elementos que son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes hay que simular.

Orientación a objetos: Es más puro pues no admiten funciones ni variables globales, todo el código y datos han de especificarse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

Orientación a componentes: Su sintaxis incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas.

Seguridad de tipos: Incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente.

Eficiencia: El código incluye numerosas restricciones para alcanzar su seguridad y no permite el uso de punteros. A diferencia de Java, en el mismo es posible saltarse dichas restricciones manipulando objetos a través de punteros.

1.2.3 ASP.NET

Es un marco de trabajo de programación generado en Common Language Runtime que puede utilizarse en un servidor para generar eficaces aplicaciones Web. Ofrece varias ventajas importantes acerca de los modelos de programación Web anteriores: [6]

- **Mejor rendimiento:** Puede aprovechar las ventajas del enlace anticipado, la compilación just-in-time, la optimización nativa y los servicios de caché desde el primer momento. Esto supone un incremento espectacular del rendimiento antes de siquiera escribir una línea de código.
- **Compatibilidad con herramientas de primer nivel:** El marco de trabajo de ASP.NET se complementa con un diseñador y una caja de herramientas muy completos en el entorno integrado de programación (Integrated Development Environment, IDE) de Visual Studio.
- **Eficacia y flexibilidad:** La biblioteca de clases de .NET Framework, la Mensajería y las soluciones de Acceso a datos se encuentran accesibles desde el Web de manera uniforme. ASP.Net es también independiente del lenguaje, por lo que puede elegir el que mejor se adapte a la aplicación o dividirla en varios lenguajes.
- **Simplicidad:** Facilita la realización de tareas comunes, desde el sencillo envío de formularios y la autenticación del cliente hasta la implementación y la configuración de sitios.
- **Facilidad de uso:** Emplea un sistema de configuración jerárquico, basado en texto, que simplifica la aplicación de la configuración al entorno de servidor y las aplicaciones Web. No se requiere el reinicio del servidor, ni siquiera para implementar o reemplazar el código compilado en ejecución.
- **Escalabilidad y disponibilidad:** El motor de tiempo de ejecución de ASP.NET controla y administra los procesos de cerca, por lo que si uno no se comporta adecuadamente (filtraciones,

bloqueos), se puede crear un proceso nuevo en su lugar, lo que ayuda a mantener la aplicación disponible constantemente para controlar solicitudes.

- **Posibilidad de personalización y extensibilidad:** Permite insertar código en el nivel adecuado, siendo posible extender o reemplazar cualquier subcomponente del motor de tiempo de ejecución de ASP.NET con su propio componente escrito personalizado.
- **Seguridad:** Con la autenticación de Windows integrada y la configuración por aplicación, se puede tener la completa seguridad de que las aplicaciones están a salvo.

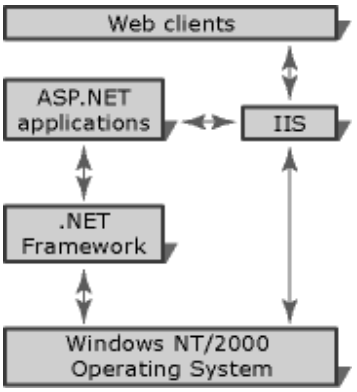


Figura 1 Arquitectura ASP.NET

Con la aparición del comercio electrónico se ha producido un incremento de la complejidad del desarrollo de este tipo de sistemas, lo cual supone un conjunto de desafíos para los desarrolladores, de los cuales pudiera citarse:

- Implementación de interfaces Web enriquecidas.
- Separación del cliente y el servidor.
- Ejecución sin control de estado.
- Capacidades del cliente desconocidas.
- Complicaciones en el acceso a datos.
- Complicaciones con la escalabilidad.

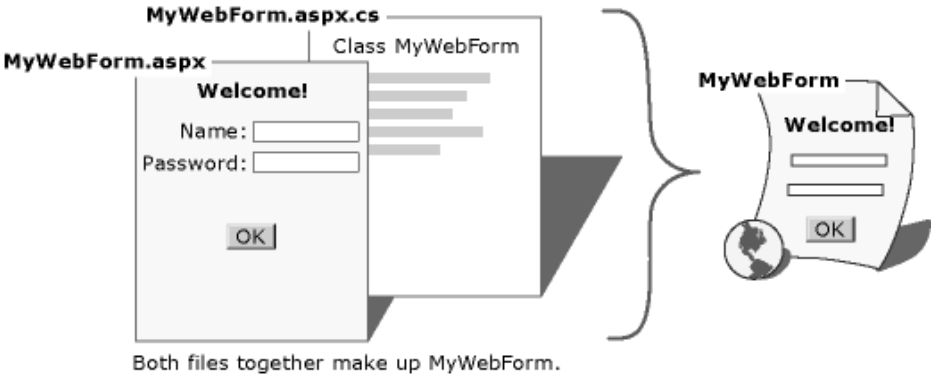


Figura 2 WebForms

La plataforma ASP.NET asume estos retos proporcionando a los desarrolladores las siguientes características:

- **Modelo de objetos intuitivo y consistente:** El marco de trabajo de las páginas ASP.NET presentan un modelo de objetos que permite concebir a los formularios como unidades, no como piezas separadas en el cliente y en el servidor. Con este modelo, se programan las páginas en una forma más intuitiva que en las aplicaciones Web tradicionales, incluyendo la capacidad de establecer propiedades para los elementos del formulario y responder a eventos. Por otro lado, los controles del servidor de ASP.NET son una abstracción del contenido físico de una página HTML y de la interacción directa entre el navegador y el servidor. En sentido general, se pueden utilizar los controles del servidor de la misma forma en que pudiera trabajarse con los controles en aplicaciones clientes sin tener que pensar en como crear el HTML para presentar y procesar los controles y su contenido.[7]
- **Modelo de programación dirigido por eventos:** Las páginas WebForms traen a las aplicaciones Web el familiar modelo de escribir manipuladores para eventos que ocurran tanto en el cliente como en el servidor. El marco de trabajo de ASP.NET abstrae este modelo de tal forma que el mecanismo subyacente de captura del evento en el cliente, su transmisión al servidor y la llamada a método apropiado es automática y transparente para el programador. El resultado es una clara y fácil estructura de código que soporta desarrollo dirigido por eventos.[8]

- **Administración de estado intuitiva:** El marco de trabajo de ASP.NET automáticamente manipula la tarea de mantener el estado de la información específica de la aplicación. Esta es llevada a cabo sin un uso intensivo de los recursos del servidor y puede ser implementada con o sin el envío de cookies al navegador.
- **Aplicaciones independientes del navegador:** ASP.NET permite la creación de toda la lógica de la aplicación en el servidor, eliminado la necesidad de producir código para diferentes navegadores. Sin embargo, este aún permite que automáticamente se tome ventaja de características específicas de los diferentes navegadores mediante la escritura de código del lado del cliente para mejorar el rendimiento. [9]

1.2.4 Modelación de aplicaciones Web sobre la plataforma ASP.NET

Para dar respuesta al problema de la modelación de aplicaciones Web aparece las extensiones de UML para Web propuestas por Jim Conallen en 1999. Esta iniciativa se basa en las características de UML de permitir extensiones del lenguaje mediante la utilización de valores etiquetados, estereotipos y restricciones para dotar a los diagramas de una nueva semántica propia del problema que se está modelando. [10]

Estas exenciones parten de la separación entre componentes en el cliente y componentes en el servidor. De ahí la aparición de sus dos conceptos más importantes, la página cliente y la página servidora. A partir de esta idea aparecen los demás elementos de la extensión: Formularios, Frameset, Target, Script, XML, entre otros.

Como se vio en la sección anterior una de las ideas más innovadoras de al plataforma ASP.NET es la unificación en una sola unidad conceptual, el WebForm, de la lógica que corre tanto en el cliente como en el servidor, haciendo transparente al desarrollador la gestión de los eventos y del estado. Esto hace posible que las aplicaciones Web puedan modelarse tal y como se modelan las aplicaciones tradicionales basadas en ventanas, sin tener que utilizar las extensiones propuestas por Conallen que describen a las páginas cliente y servidora.

Basándose en lo expresado anteriormente, en el presente trabajo se ha optado por modelar las páginas como clases de interfaz tradicionales, sin hacer la distinción entre páginas clientes y páginas servidoras. Con esto se simplifica el diseño modelándose los elementos tal como serán utilizados en la etapa de implementación. Además, se ha adoptado como política del proyecto global un conjunto de medidas para aprovechar al máximo las abstracciones de la plataforma ASP.NET, las cuales son:

- Eliminar la programación en el cliente. Aunque esto supone un deterioro del desempeño de la aplicación, elimina la necesidad de escribir código específico para cada uno de los navegadores más utilizados.
- Utilización intensiva de controles en el servidor. Esta medida apoya la anterior y permite dejarle a estos la generación del código específico para cada navegador
- No utilización de elementos ajenos a la plataforma como applets java y componentes ActiveX en el cliente.
- No utilización de marcos.

1.2.5 Programación Multicapas

Siguiendo la filosofía del modelo actual de desarrollo del software, para la realización del sistema se propone organizar los elementos de la aplicación en componentes independientes buscando alcanzar una mayor efectividad a la hora de administrarlos.

Para ello se seguirá los fundamentos de la programación en múltiples capas ya que esta además de facilitar una administración eficiente de los componentes que la integran, proporciona rapidez a todas las funcionalidades Cliente-Servidor y la magnitud de la aplicación lo exige. Tal y como plantea la arquitectura de esta tecnología, los componentes primarios de la aplicación serán divididos y programados por separados y en tiempo de ejecución serán unidos. De forma tal que si una de las capas definidas sufre cambios, no se vean afectados el resto de las capas ni el resultado final del producto.

Se definen tres capas, tal es el caso de: Presentación, Reglas del Negocio y Acceso a Datos.

Presentación

En esta capa se diseña todo lo que constituye la interfaz gráfica y la interacción del usuario con el sistema.

Reglas del Negocio

Contiene todas las subrutinas creadas con el propósito de regular alguna acción del usuario.

Acceso a Datos

En esta capa se programa todo lo que tiene que ver con el acceso a la base de datos. Esta capa queda encargada de tomar la información de la base de datos dada una petición de la capa de Reglas del Negocio, que a su vez es generada por la capa de presentación.

Figura 3 Aplicaciones de tres capas

Ventajas del modelo

- Desarrollos paralelos (en cada capa)
- Aplicaciones más robustas debido al encapsulamiento
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica)
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad)

- Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

1.2.6 AJAX

Acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript y XML asíncronos), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de tres tecnologías ya existentes:

- **XHTML** (o **HTML**) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto **XMLHttpRequest** para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

Como el DHTML, LAMP o SPA, AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.

1.2.7 Servicios Web (Web Services)

Los Web Services son componentes software que permiten a los usuarios usar aplicaciones de negocio que comparten datos con otros programas modulares, vía Internet. Son aplicaciones independientes de la plataforma que pueden ser fácilmente publicadas, localizadas e invocadas mediante protocolos Web estándar, como XML, SOAP, UDDI o WSDL. El objetivo final es la creación de un directorio de online de Web Services, que pueda ser localizado de un modo sencillo y que tenga una alta fiabilidad. [11]

El principal objetivo que se logra, es la interoperabilidad y la integración.

1.2.7.1 Protocolos que usan los Servicios Web

XML (Extensible Markup Language): Es un metalenguaje de marcas que permite definir cómo es la información que se transmite. Esto permite una comunicación de datos entre distintos sistemas. Es la base de los Servicios Web, y a pesar de su sencillez aparente, está transformando completamente la creación y el uso de software. Es la solución a un problema de comunicación entre programas de ordenador, pues la información generalmente queda fuertemente ligada al programa con el cual fue creada, y es así como se pierde mucho tiempo en pasar de un formato de definición a otro. El contenido almacenado en un documento XML se puede transferir fácilmente a través de la red.

SOAP (Single Object Access Protocol): Es un protocolo de mensaje liviano basado en XML, usado para codificar los mensajes de Web Services antes de enviarlos por la red. Los mensajes SOAP son independientes de cualquier sistema operativo y protocolo, y pueden ser transportados usando una variedad de protocolos de internet, incluyendo HTTP, SMTP y MIME. Permite que programas que corren en diferentes sistemas operativos se comuniquen. [12]

WSDL (Web Service Description Language): Es un lenguaje en formato XML que define las operaciones que proporciona un servicio, desarrollado conjuntamente por Microsoft e IBM.

UDDI (Universal Description Discovery and Integration): Es un directorio universal de Servicios Web basado en XML, que permite publicar, localizar y utilizar servicios Web.

1.3 RUP

Se hizo uso de las herramientas de la metodología RUP (*Rational Unified Process*) para facilitar el desarrollo del sistema.

El Proceso Unificado es un proceso de desarrollo de software que contiene un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software (Figura1). Más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, tipos de organizaciones, niveles de actitud y tamaños de proyecto. Está basado en componentes, lo cuál quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas. Utiliza el *Lenguaje Unificado de Modelado* (Unified Modeling Languaje, UML) para preparar todos los esquemas de un sistema software. Garantiza la elaboración de todas las fases de un producto de software orientado a objetos. [13]

UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

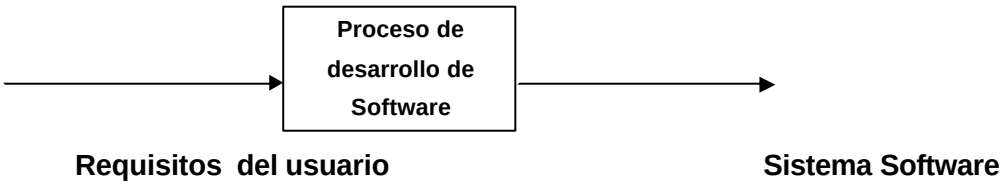


Figura 4. Un proceso de desarrollo de software.

1.3.1 Características del Proceso Unificado

Los verdaderos aspectos definitorios del Proceso Unificado, y que lo convierten en único, se resumen en tres frases clave - dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

- **Dirigido por los casos de uso:**

Teniendo en cuenta que la razón de ser de un sistema es brindar servicios a los usuarios, RUP define caso de uso como el conjunto de acciones que debe realizar un sistema para dar un resultado de valor a un determinado usuario y los utiliza tanto para especificar los requisitos funcionales del sistema, como para guiar todos los demás pasos de su desarrollo, dígase diseño, implementación y prueba.

- **Estar centrado en la arquitectura:**

La arquitectura es una vista del diseño completo con las características más importantes, dejando a un lado los detalles. Esta no solo incluye las necesidades de los usuarios e inversores, sino también otros aspectos técnicos como el hardware, sistema operativo, sistema de gestión de base de datos, protocolos de red; con los que debe coexistir el sistema. En otras palabras, la arquitectura representa la forma del sistema, la cual va madurando en su interacción con los casos de uso hasta llegar a un equilibrio entre funcionalidad y características técnicas.

- **Ser iterativo e incremental:**

La alta complejidad de los sistemas actuales hace que sea factible dividir el proceso de desarrollo en varios mini-proyectos. Cada uno de estos mini-proyecto se les denomina iteración y pueden o no representar un incremento en el grado de terminación del producto completo. En cada iteración los desarrolladores seleccionan un grupo de casos de uso, los cuales se diseñan, implementan y prueban. La planificación de iteraciones hace que se reduzcan los riesgos de los costes de un solo incremento, no sacar al mercado un producto en el tiempo previsto, mantener la motivación del equipo pues puede ver avances claros a corto plazo y que el desarrollo pueda adaptarse a los cambios en los requisitos.

1.3.2 Rational Rose

Es una herramienta para “modelado visual”, que forma parte de un conjunto más amplio de herramientas que juntas cubren todo el ciclo de vida del desarrollo de software. Permite completar una gran parte de las disciplinas (flujos fundamentales) del proceso unificado de Rational (RUP) e incluye un conjunto de herramientas de ingeniería inversa y generación de código que allanan el camino hasta el producto final.

Conclusiones

Después de haber realizado un análisis de la información recopilada sobre los diferentes sistemas de gestión, se puede concluir que ninguno de los estudiados se adecua a las características requeridas por los objetivos del sistema propuesto.

Con las características expuestas sobre servicios Web, se puede concluir que son parte de la revolución informática de la nueva generación de aplicaciones que trabajan en colaboración y el objetivo principal que se logra con el uso de mismos es la interoperabilidad y la integración, pues permiten que las aplicaciones compartan información e invoquen funciones de otros sistemas. Lo cual será utilizado para garantizar la integración del conjunto de aplicaciones que forman el Sistema de Información de Laboratorios Clínicos.

Como tecnología, las arquitecturas multicapas proporcionan una gran cantidad de beneficios en soluciones flexibles y fiables para resolver complejos problemas inmersos en cambios constantes además permiten a los componentes de negocio correr en una LAN, WAN o Internet. Esto significa, que cualquiera con un ordenador y conexión a la Red posee toda la funcionalidad que tendría si se encontrase delante de su sistema de escritorio, de ahí que se utilicen estas características para el desarrollo de la aplicación.

Finalmente para el desarrollo del sistema se utilizarán las siguientes tecnologías por las características fundamentadas con anterioridad: la plataforma .NET de Microsoft (C# y ASP .NET), como Gestor de Base de Datos PostgreeSQL, como metodología y aplicación para el diseño RUP y Racional Rose.

CAPITULO 2

Descripción y análisis de la solución propuesta

Introducción

En este capítulo se describe la construcción de la solución propuesta a partir del análisis de dicha solución llevado a nivel de casos de usos. Se construyen los diagramas de clases del diseño, el modelo de datos y se definen los principios de diseño, estándares de codificación y se construye el modelo de despliegue.

2.1 Valoración del diseño propuesto por el analista

Para tener una idea clara de la solución propuesta, primero es necesario entender el contexto donde se desarrolla el sistema. Mediante el diagrama de casos de uso y la descripción de cada uno de ellos, se definen los límites del sistema y las relaciones entre el sistema y el entorno y a su vez cada caso de uso responde a uno o varios requerimientos funcionales del sistema.

2.1.1 Requisitos Funcionales

- ✓ **Reservar examen:** El doctor, dado la necesidad de realizar un análisis, le solicita una serie de exámenes (que conforman una solicitud) para eso debe introducir ciertos datos (fecha hora de solicitud, Nombre del paciente que debe estar inscrito, nombre de quien reserva los exámenes).
- ✓ **Recepcionar solicitud**: Se introducen los análisis que se le van a realizar (hay que tener en cuenta la existencia o capacidad del laboratorio para relazar el servicio solicitado en ese momento) y con ellos se confecciona una Solicitud de Análisis que puede tener una o varias Ordenes de Análisis, se informa de el tiempo de demora aproximado máximo de todos los análisis (De ser un examen que se realiza otro centro se informa a el interesado).
- ✓ **Tomar Muestra:** El encargando de las muestras toma las muestras, evalúa su calidad y si es procedente toma la hora en que termina la extracción, y alguna observación especial, se define si

se almacenara o no y de no poder proceder con la muestra, por mala calidad realizar el procedimiento correspondiente pare tomar nuevamente la muestra.

- ✓ **Almacenar Muestra:** Consiste en definir el tiempo y las condiciones de almacenamiento de una muestra; temperatura, luz, Envase, reactivo.
- ✓ **Informar de nueva solicitud:** Notificar a el interesado sobre la existencia de una nueva solicitud de análisis.
- ✓ **Buscar Historia Clínica:** Buscar si la persona está inscrita en el hospital.
- ✓ **Autentificarse:** Garantizar el acceso a un usuario a cada una de sus obligaciones y dependencias y restringir el acceso a lugares que no le pertenecen.
- ✓ **Imprimir:** Todos los usuarios podrán acceder al servicio de impresión en cada momento.
- ✓ **Emitir resultados:** El personal que realiza el análisis podrá acceder a sus pendientes y llenarlo en cualquier momento.
- ✓ **Modificar Resultado:** Modificar un resultado o completarlo.
- ✓ **Buscar Solicitud:** buscar una solicitud de análisis que puede incluir varios análisis o solo uno.
- ✓ **Buscar Análisis:** Buscar una orden de análisis especifica.

2.1.2 Requerimientos no Funcionales

- ✓ **Requerimientos de apariencia o interfaz externa:** El sistema debe tener un ambiente amigable y entendible para los usuarios finales, de forma tal que no les sea muy complicado utilizar el software. No debe utilizarse tecnología de *frames.* Cada página no debe exceder los 500 Kb en imágenes. *(Ver especificaciones del diseño de la interfaz).*
- ✓ **Requerimientos de usabilidad:** La aplicación debe cumplir con los principales principios de usabilidad, debe brindarse comodidad a la hora de acceder a las diferentes funcionalidades que brinda la aplicación mediante teclas de acceso rápido, la navegabilidad debe no debe ser muy compleja, todas las funcionalidades deben ser rápidamente accesibles por el usuario.
- ✓ **Requerimientos de rendimiento:** El tiempo de respuesta de una petición al servidor deber rápido para la toma de decisiones.
- ✓ **Requerimientos de soporte:** Se le debe dar mantenimiento periódicamente a los servidores de bases de datos controlando la integridad de la información.
- ✓ **Requerimientos de portabilidad:** El sistema debe ser capaz de actualizarse a sí mismo.

- ✓ **Requerimientos de seguridad y privacidad:** La información debe transmitirse de manera segura, se debe garantizar la seguridad a todos los niveles (Interfaz, negocio y Acceso a datos) restringiendo las funcionalidades mediante roles de usuarios garantizando que la información sea accesible al usuario autorizado.
- ✓ **Requerimientos de confiabilidad:** La información debe transmitir a través de canales seguros. Se debe chequear la integridad de los datos.
- ✓ **Requerimientos de ayudas y documentación en línea:** Se debe brindar una interfaz amigable que explique las diferentes funcionalidades con que cuenta el sistema de manera rápida, además los manuales de usuario y toda la documentación actualizada de cada módulo de la aplicación.
- ✓ **Requerimientos de hardware:** Requerimientos para una estación de trabajo: 256Mb RAM (Recomendado 512Mb), 1GHz, 10Gb HDD. Requerimentos para un servidor: 512Mb RAM (Recomendado 1Gb RAM o superior), 1GHz o superior, 60Gb HDD
- ✓ **Requerimientos de software:** El sistema debe correr en sistemas operativos Windows 98 o superior y sistemas Unix, Linux. Para sistemas Windows se deber tener instalado Microsoft Framework 2.0 y en sistemas Linux la plataforma Mono 1.2 o superior.
- ✓ **Restricciones en el diseño y la implementación:** El sistema será implementado utilizando como lenguaje de programación del lado del cliente javascript auxiliándose de la tecnología AJAX. Se utilizará la plataforma de desarrollo Microsoft .NET que brinda una gama de facilidades en su entorno y que da la posibilidad de utilizar el lenguaje C Sharp del lado del servidor. Se utilizará además un grupo de bibliotecas de clases definidas, dentro de ellas se encuentran Hermes7 para la comunicación HL7 y NpgSql para la conexión al servidor de bases de datos. La comunicación de de las terminales clientes con el servidor será a través de conexiones de fibra óptica.

2.1.3 Concepción general del negocio

El negocio comienza cuando el doctor en su consulta decide hacer un examen de laboratorio. El realiza una solicitud de análisis en la que incluye datos del Paciente (nombre, HC, etc.) luego escoge los exámenes a realizar, y también el tipo de muestra a tomar, la fecha y hora, su firma y tal vez algún otro comentario adicional. El paciente se persona en el laboratorio para la toma de muestra (puede que alguien traiga su muestra) la recepcionista recoge la fecha y hora en la que se presenta el paciente (también puede ser la muestra en todo caso de aquí en adelante se usará paciente refiriéndose a la persona que

trae la muestra), si el doctor no escogió el tipo de la muestra ella se encarga de escogerla, luego se procede a tomar la muestra este proceso puede ser de 2 formas el paciente trae la muestra ya tomada o que haya que tomársela, en todo caso los datos que se toman de la muestra son los mismos.

Se toma la hora (la fecha no es necesaria ya fue tomada por la recepcionista), un estado de muestra; que puede ser vacío en caso de estar bien. Si la muestra no va a ser procesada en el momento decir la razón, el tiempo que estará guardada, y especificar las condiciones en que se guardará. Luego de esto la muestra pasa al técnico, que se encarga de realizar los análisis; el técnico recibe una serie de Solicitudes ordenadas y con un número que las identifica dichas solicitudes serán realizadas en dependencia de la prioridad de las mismas pero teniendo en cuenta la cola de solicitudes que se tengan. El técnico llega a un resultado de el análisis y luego de que termina en responsable de área lo libera (en caso de laboratorios pequeños el propio técnico puede liberarlos).

El Proceso de liberación consiste en; Si ya se termino definitivamente liberarlo y eliminar la muestra, si luego de tener el resultado el responsable de liberarlos considera que es necesario realizar otros análisis adiciónales los escoge procede a asignarlos a el persona correspondiente (si para realizar estos análisis es necesario que se le tomen muestras al paciente nueva mente se procede a citarlo). En caso de que el responsable considere que la muestra no se debe liberar por alguna razón; si se considera que hubo malos procedimientos con la muestra o para realizar el análisis nuevamente se procede a tomar la muestra nuevamente al paciente, si le parece que el resultado del examen no es correcto procede a realizar el análisis nuevamente (con la misma muestra de ser posible, si no se solicita una nueva).

2.1.3.1 Descripción de los casos de usos del sistema

Definición de los actores

Actores	Justificación
Doctor	Interactúa con el sistema. Introduce los datos del paciente en el mismo para emitir una solicitud de análisis.
Recepcionista	Interactúa con el sistema. Se encarga de

	recepcionar la solicitud de análisis una vez emitida por el doctor. Gestiona la información de paciente cuando este se presenta en el laboratorio.
Actor con permiso	Es una generalización de los actores Doctor, Recepcionista y Técnico que realizan en común la búsqueda de una solicitud de análisis de un paciente determinado.
Encargado de muestra	Interactúa directamente con el sistema. Es el encargado de tomar los datos de la muestra y poner le datos de almacenaje en caso de ser necesario.
Técnico	Interactúa directamente con el sistema. Se encarga de emitir un resultado una vez analizada la muestra. Además se encarga de buscar un análisis o modificar un resultado en caso que sea necesario.
Almacenero	Interactúa directamente con el sistema. Se encarga del control de materiales del laboratorio.
Usuario	Es una generalización de los actores Técnico y Jefe de Área, que realizan en común la búsqueda de un análisis y la modificaron o emisión de un resultado.
Jefe de área	Interactúa directamente con el sistema. Se encarga de la liberar un análisis y de incluir mas análisis en una solicitud (de ser necesario) una vez emitido un resultado.

CUN Emitir Solicitud

Caso de uso	
CU-1	Emitir Solicitud
Propósito	Emitir una solicitud de análisis.
Actores : Doctor **Resumen:** El caso de uso inicia cuando el paciente se presenta con el doctor y este decide emitir una solicitud de análisis. El doctor le toma todos los datos necesarios y luego emite la solicitud de análisis.	
Referencias	RF1

Acción del actor	**Respuesta del sistema**
1- El doctor entra al sistema, específicamente a la sección de laboratorios clínicos. 3- El doctor introduce los datos solicitados. 4- Presiona el botón Aceptar.	2- El sistema muestra la página para introducir datos del paciente y análisis a realizar. 5- El sistema verifica que los campos obligatorios no estén vacíos. 6- El sistema emite la solicitud.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	5.1- Si alguno de los campos obligatorios están vacíos, el sistema emite un mensaje para que llene dichos campos.
Prioridad: Crítico	

CUN Recepcionar Solicitud Análisis

Caso de uso

CU-2	Recepcionar Solicitud de análisis
Propósito	Recepcionar una solicitud de análisis, o sea cuando el paciente se presenta en el laboratorio para realizarse su análisis, la recepcionista verifica que se haya emitido su solicitud de análisis.
Actores: Recepcionista	
Resumen: El caso de uso inicia cuando el paciente se presenta en el laboratorio (primeramente en la recepción) para la realización del análisis emitida por el doctor. La recepcionista verifica que la solicitud haya sido emitida, o sea que busca esta solicitud y una vez encontrada le indica al paciente como proceder.	
Referencias	RF2

Acción del actor	**Respuesta del sistema**
1- La recepcionista entra al sistema. 3- La recepcionista busca la emisión del paciente.	2- El sistema muestra la bandeja de entrada de Recepcionar Solicitud, donde aparecen todos los análisis emitidos por el doctor.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
3.1- La recepcionista selecciona la opción Buscar Solicitud de Análisis. 5- La recepcionista anota los datos.	4- El sistema muestra una ventana con los datos necesarios para la búsqueda. 6- El sistema realiza la búsqueda.
Prioridad: Crítico.	

CUN Buscar Solicitud

Caso de uso	
CU-6	Buscar Solicitud
Propósito	Buscar una solicitud.
Actores: Doctor, Recepcionista, Jefe de área.	
Resumen: Este caso de uso inicia cuando el doctor, recepcionista o Jefe de área necesitan buscar una solicitud de análisis de un paciente por cualquier motivo. El caso de uso finaliza cuando se encuentra dicha solicitud.	

Referencias	RF8
Acción del actor	**Respuesta del sistema**
1- El actor decide hacer la búsqueda de una solicitud. 3- El actor introduce los datos y presiona el botón Buscar.	2- El sistema le muestra la ventana con los datos que debe introducir. 4- El sistema busca la solicitud y la muestra.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	4.1- El sistema no encuentra ninguna solicitud de análisis correspondiente a los datos introducidos.
Prioridad: Crítica	

CUN Buscar Análisis

Caso de uso	
CU-7	Buscar Análisis
Propósito	Buscar un análisis.
Actores: Técnico y Jefe de Área. **Resumen:** El caso de uso inicia cuando el técnico o el jefe de área necesitan buscar un análisis para emitir un resultado una vez analizada la muestra. El caso de uso finaliza cuando se encuentra dicho análisis.	
Referencias	RF9
Acción del actor	**Respuesta del sistema**
1- El actor decide buscar un análisis. 3- El actor introduce los datos y presiona el botón buscar.	2- El sistema muestra la ventana con los datos que debe introducir. 4- El sistema busca el análisis y lo muestra.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	4.1- El sistema no encuentra el análisis.
Prioridad: Crítica	

CUN Liberar Resultado

Caso de uso	
CU-8	Liberar Resultado
Propósito	Liberar un resultado después de verificar que no tenga errores.
Actores: Jefe de Área.	
Resumen: El caso de uso inicia cuando el jefe de área verifica que el resultado del análisis no sea erróneo. De no ser erróneo los libera. El caso de uso finaliza cuando este resultado le llega al doctor que emitió la solicitud y a la recepcionista del laboratorio	
Referencias	RF6
Acción del actor	**Respuesta del sistema**
1- El jefe de área selecciona la opción liberar análisis. 3- El jefe de área selecciona el análisis a liberar. 5- El jefe de área introduce los datos necesarios y presiona el botón Liberar.	2- El sistema muestra una bandeja de entrada con los análisis pendientes de liberación. 4- El sistema muestra una ventana con datos de la solicitud de análisis, datos de la muestra, y campos en los que el jefe de área debe introducir el resultado del análisis otras especificaciones. 6- El sistema libera el este resultado.
Flujo alternativo 1	
Acción del actor	**Respuesta del sistema**
5.1- El jefe de área presiona el botón No Válido. 7- El jefe de área selecciona la opción Volver a realizar análisis y presiona el botón enviar.	6.1- El sistema muestra una ventana para especificar que debe hacerse. La ventana muestra las siguientes opciones: - Volver a realizar análisis. - Volver a analizar muestra - Realizar otro análisis. 8- El sistema envía una notificación a la recepción.
Prioridad: Critica	
Flujo alternativo 2	
Acción del actor	**Respuesta del sistema**

7.1- El jefe de área selecciona la opción Volver a analizar muestra.	8.1- El sistema envía una notificación al técnico.
Flujo alternativo 3	
Acción del actor	**Respuesta del sistema**
7.1- El jefe de área selecciona la opción Realizar otro análisis.	8.1- El sistema envía una notificación a al Recepción.

CUN Modificar Resultado

Caso de uso		
CU-10	Modificar Resultado	
Propósito	Modificar el resultado de un análisis.	
Actores: Técnico y Jefe Area		
Resumen: El caso de uso se inicia cuando se desea cambiar el resultado de análisis, ya sea porque se hayan equivocado a la hora de poner el resultado o porque se haya vuelto a realizar el análisis.		
Referencias	RF11	
Acción del actor		**Respuesta del sistema**
(6)		(7)
Flujo alternativo (8)		
Acción del actor		**Respuesta del sistema**
(9)		(10)
Puntos de extensión.		
(11)		

CUN Control de materiales

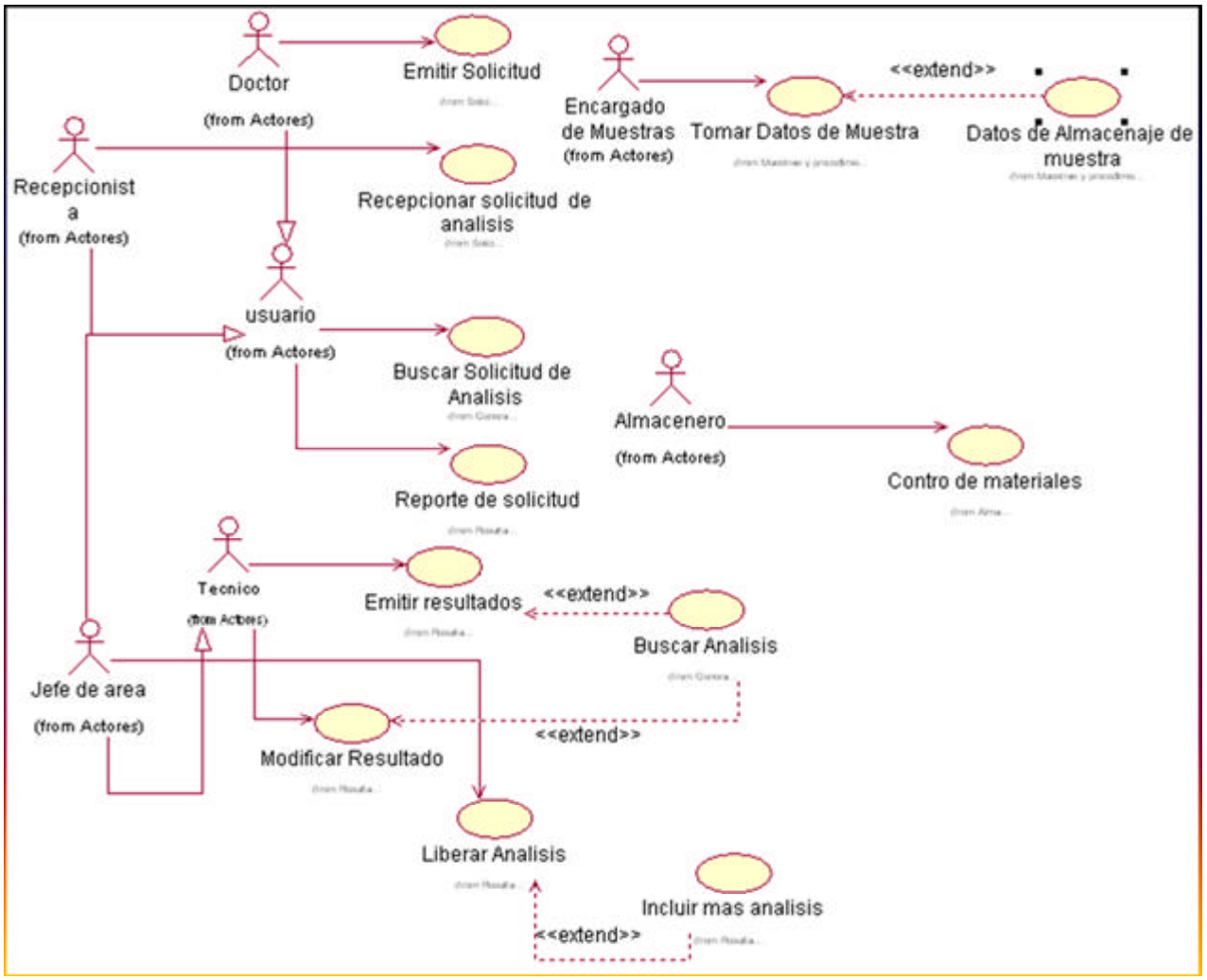
Caso de uso	
CU-11	Control de materiales del laboratorio
Propósito	Llevar un control de los materiales del laboratorio.
Actores: Almacenero	

Resumen: El caso de uso se inicia cuando el almacenero necesita controlar los materiales que salen del almacén, para llevar un control estadístico sobre dichos materiales.	
Referencias	RF10
Acción del actor	**Respuesta del sistema**
(6)	(7)
Flujo alternativo (8)	
Acción del actor	**Respuesta del sistema**
(9)	(10)
Puntos de extensión.	
(11)	

CUN Emitir Resultado

Caso de uso	
CU-12	Emitir Resultado
Propósito	Emitir el resultado de un análisis después de analizarlo.
Actores: Técnico o Jefe de Área.	
Resumen: El caso de uso inicia cuando se analiza la muestra del paciente. El técnico o el jefe de área (ambos pueden analizar la muestra), después de analizar la muestra emiten un resultado. Dicho resultado se queda en el laboratorio hasta que se verifique que no es erróneo.	
Referencias	RF5
Acción del actor	**Respuesta del sistema**
1- El actor usuario decide emitir el resultado de un resultado, para ello presiona.	
Flujo alternativo (8)	
Acción del actor	**Respuesta del sistema**
(9)	(10)
Puntos de extensión.	
(11)	

2.1.4 Diagrama de casos de usos del sistema



2.1.5 Conclusión sobre el diseño propuesto por el analista

Los artefactos resultantes en los flujos de trabajos de requisitos y análisis crean una base sólida para el sistema de construcción, generando varios prototipos de confrontación con el cliente para evitar malos entendidos en las siguientes iteraciones.

2.2 Construcción de la solución propuesta

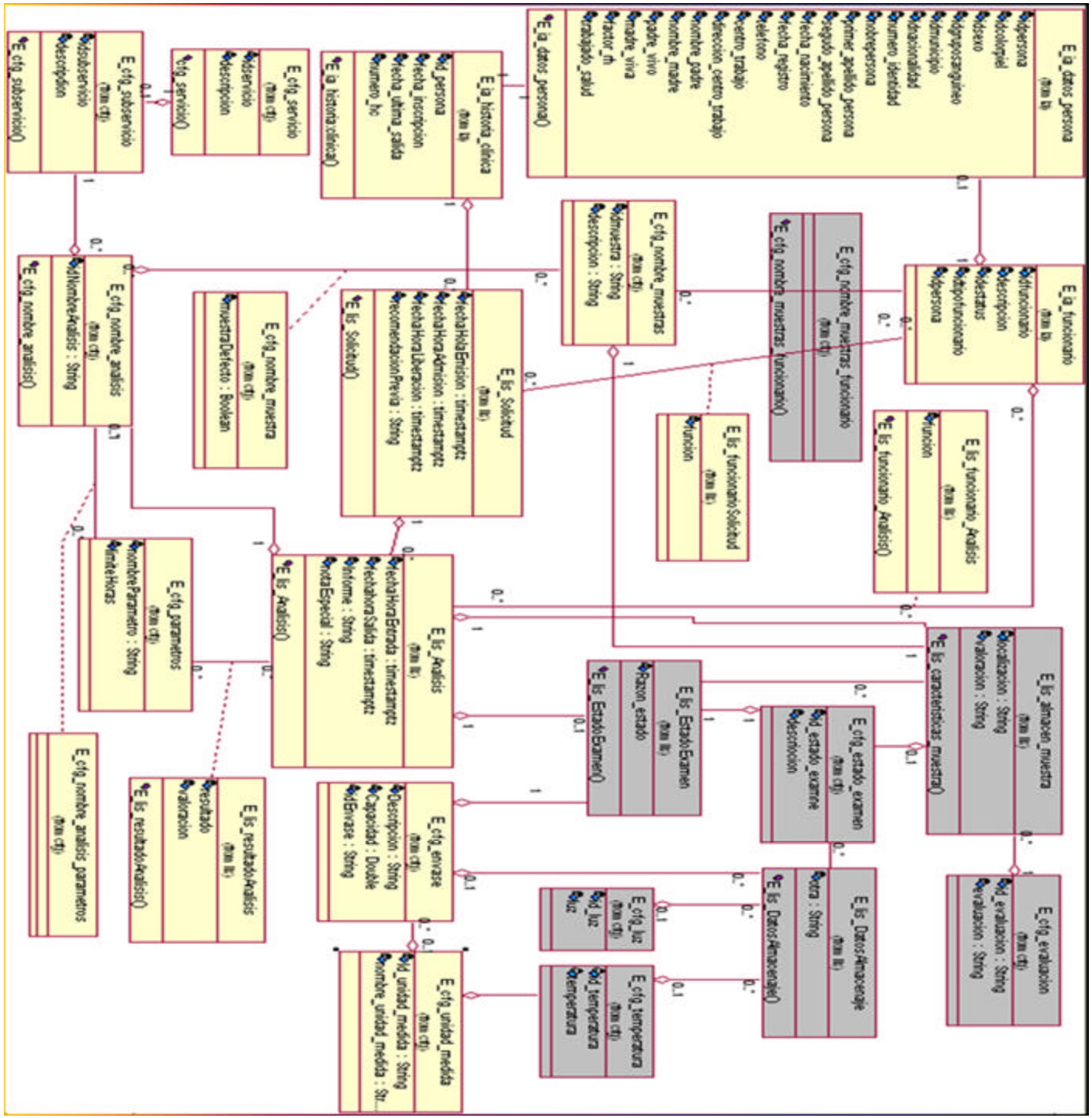
El diseño es la parte del proceso de desarrollo de software que tiene como objetivo principal definir la estructura del sistema en estudio, siempre basándose en los requisitos funcionales que fueron seleccionados en las etapas anteriores.

A continuación se muestran los principales artefactos UML obtenidos en los flujos de trabajo de diseño e implementación según RUP. Para su estudio más detallado el diagrama de clases se estructuró por paquetes, obteniendo como resultado el diagrama de clases persistentes y su modelo lógico de datos. Finalizando, se presentarán el modelo de implementación mediante el diagrama de componentes y de despliegue que resultaron del diseño realizado de cada uno de los casos de uso del sistema.

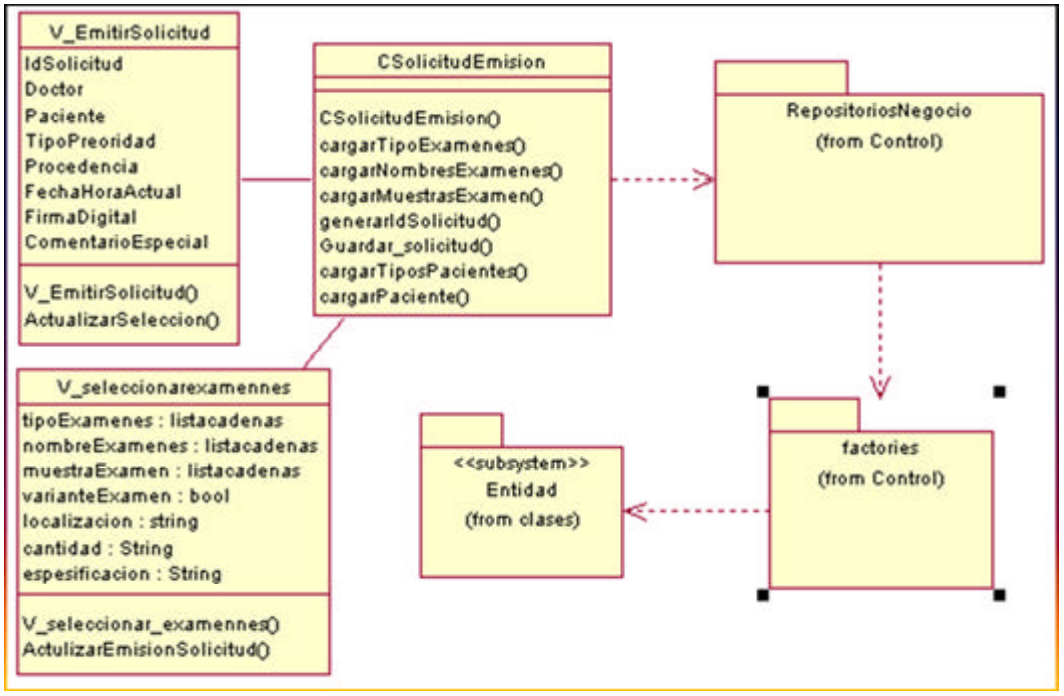
Una de las características más relevantes de la notación UML es su capacidad para absorber nueva semántica sin romper su lógica interna. La necesidad de implementar Servicios Web a través de complejas arquitecturas con múltiples capas de componentes y una gran dispersión geográfica de nodos, ha supuesto todo un reto al abordar su modelado y especificación.

Alguien dijo alguna vez:
"UML es una caja de herramientas, como un cajón de sastre, no tienes por qué usarlas todas y ni siquiera tienes por qué usarlas para lo que está especificado que se usen".

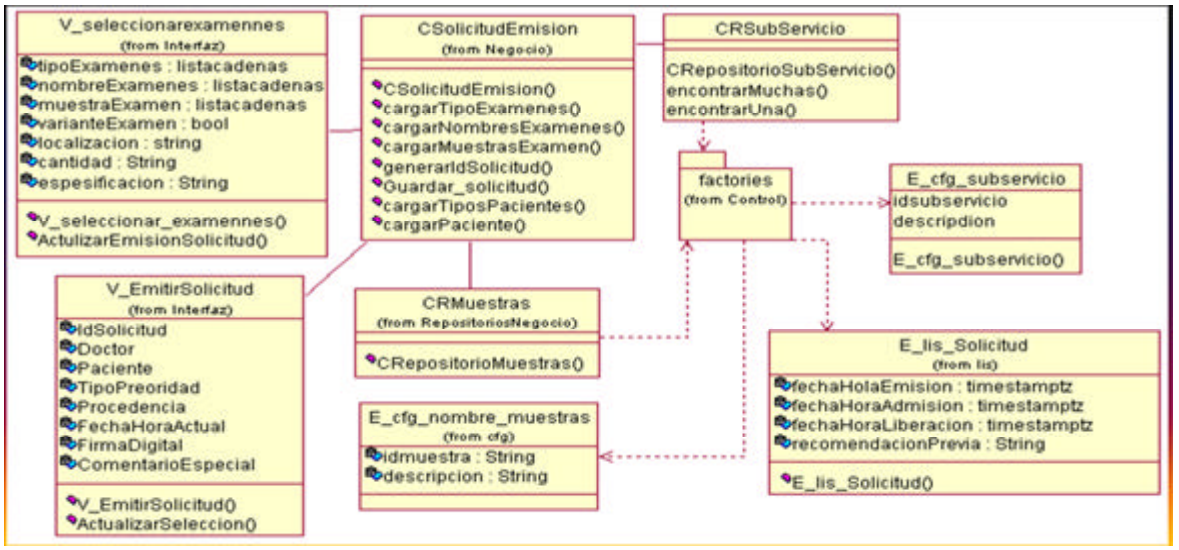
2.2.1 Diagrama de clases del diseño



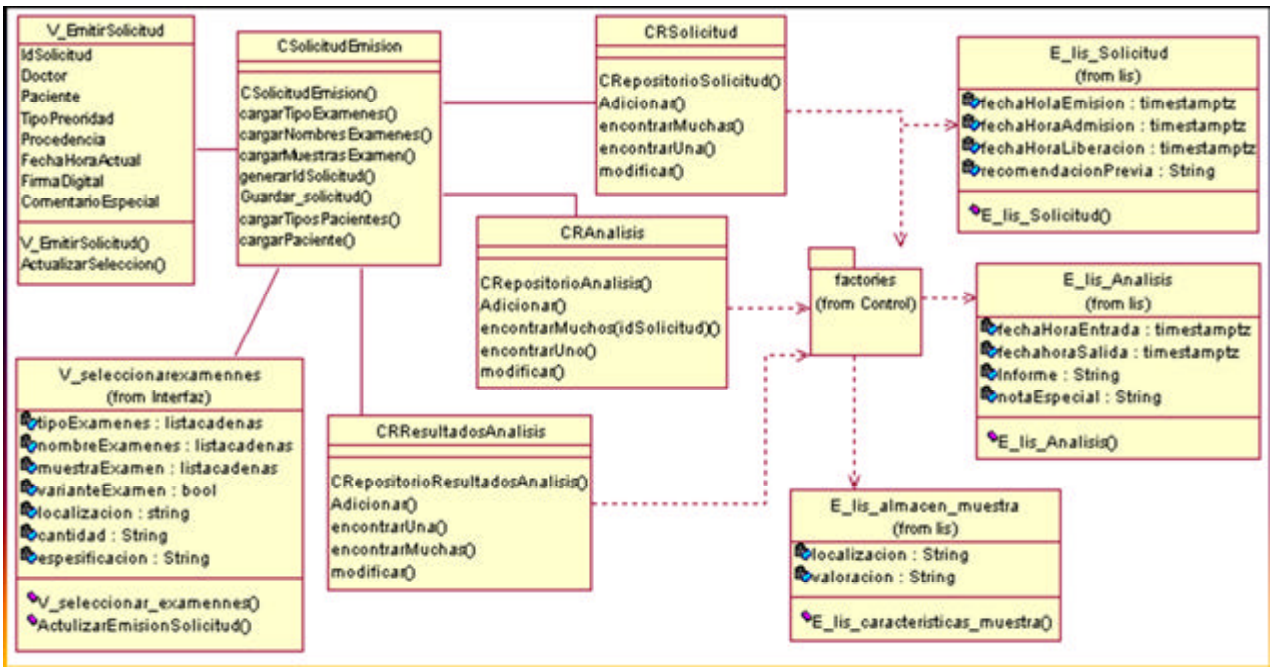
2.2.2 RCU Emitir Solicitud



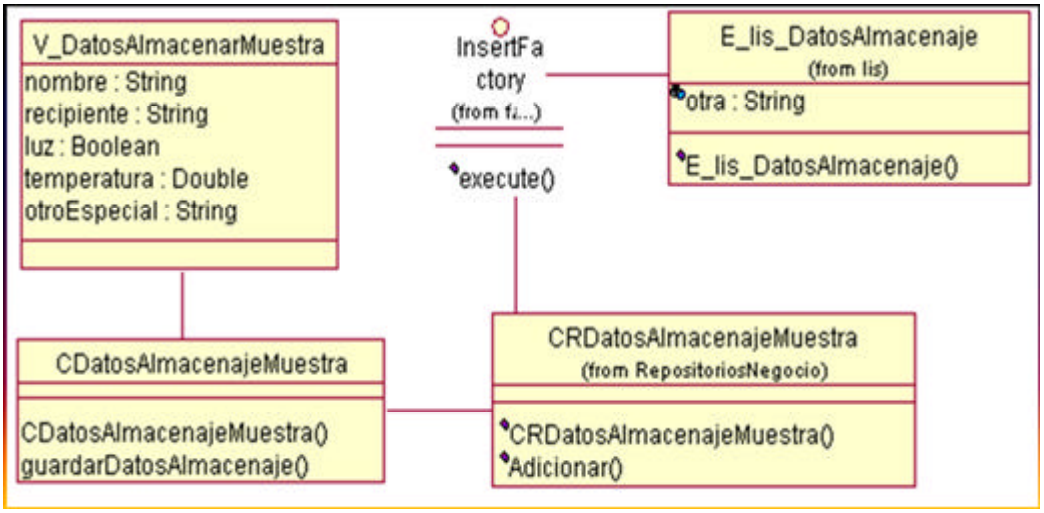
RCU Emitir solicitud (Seleccionar Examen)



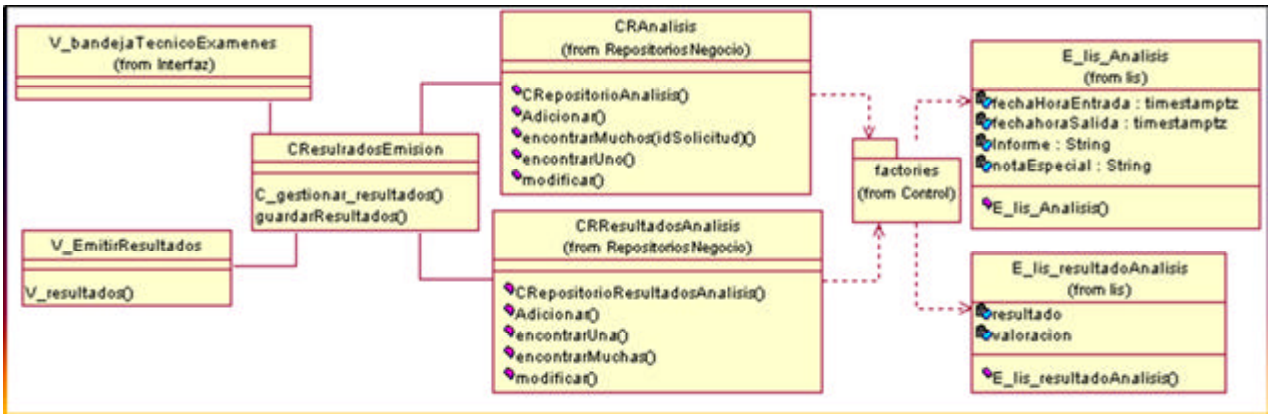
RCU Emitir Solicitud (Enviar Solicitud)



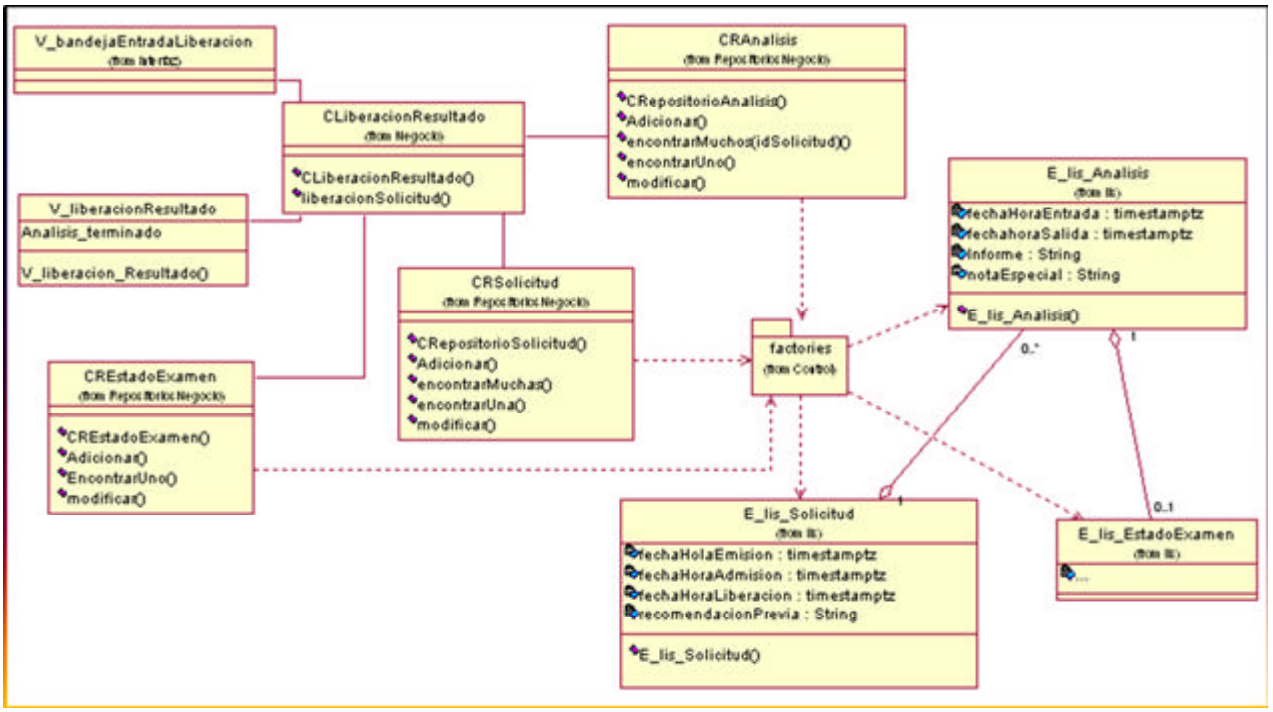
2.2.3 RCU Almacenar Muestra



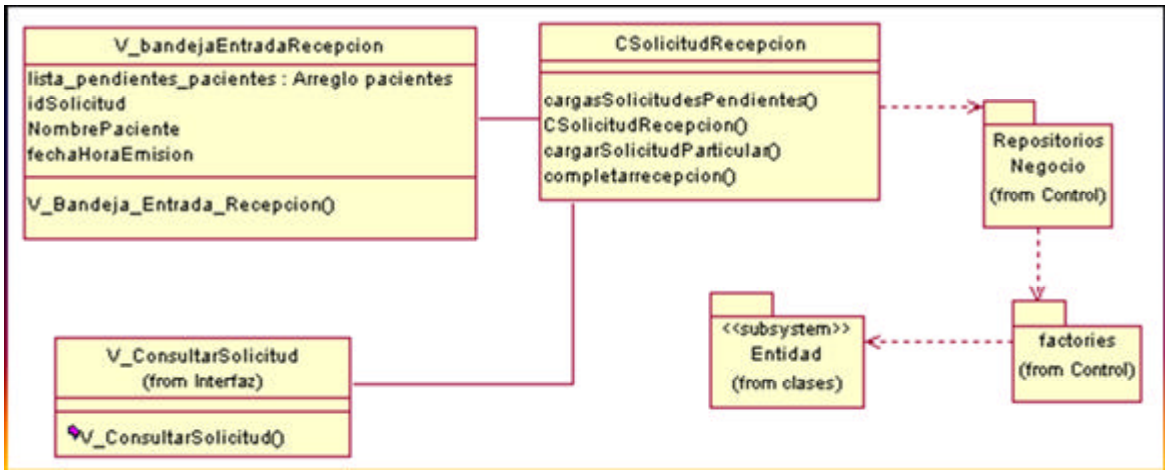
2.2.4 RCU Emitir Resultado



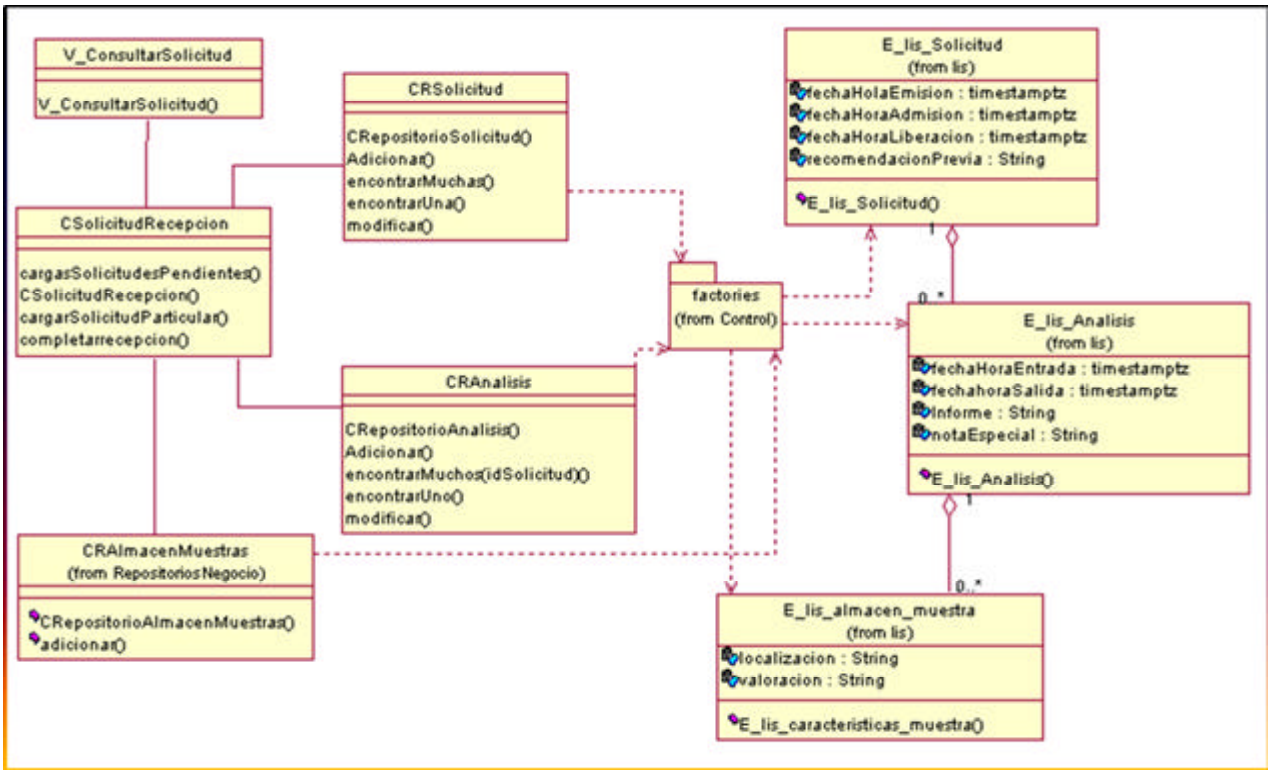
2.2.5 RCU Liberar Resultados



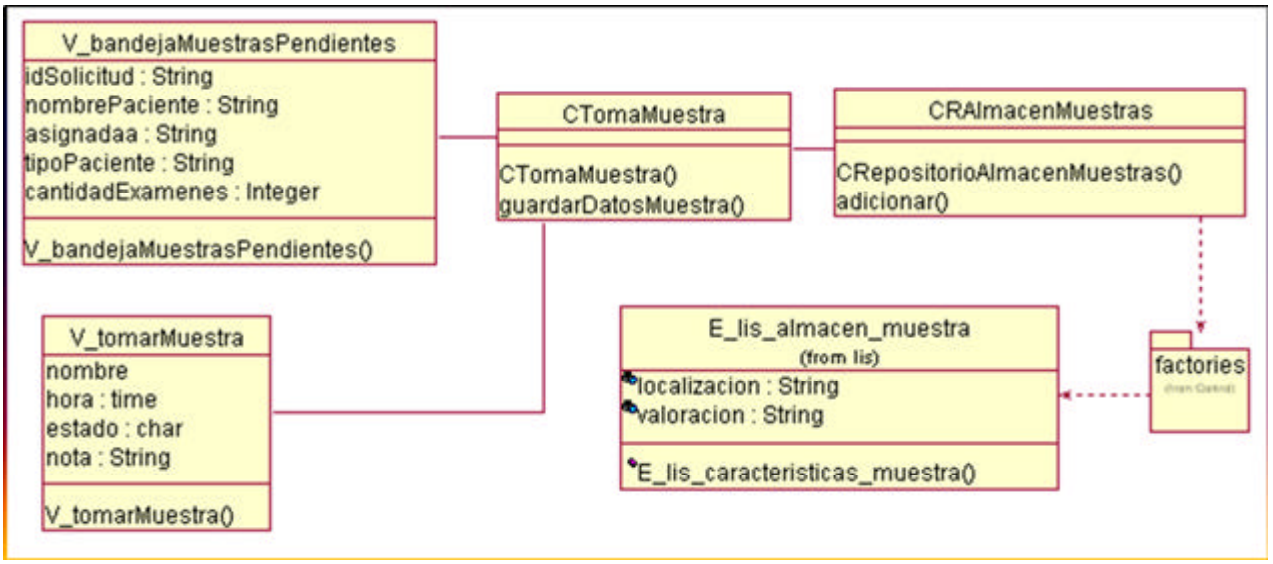
2.2.6 RCU Recepcionar Solicitud



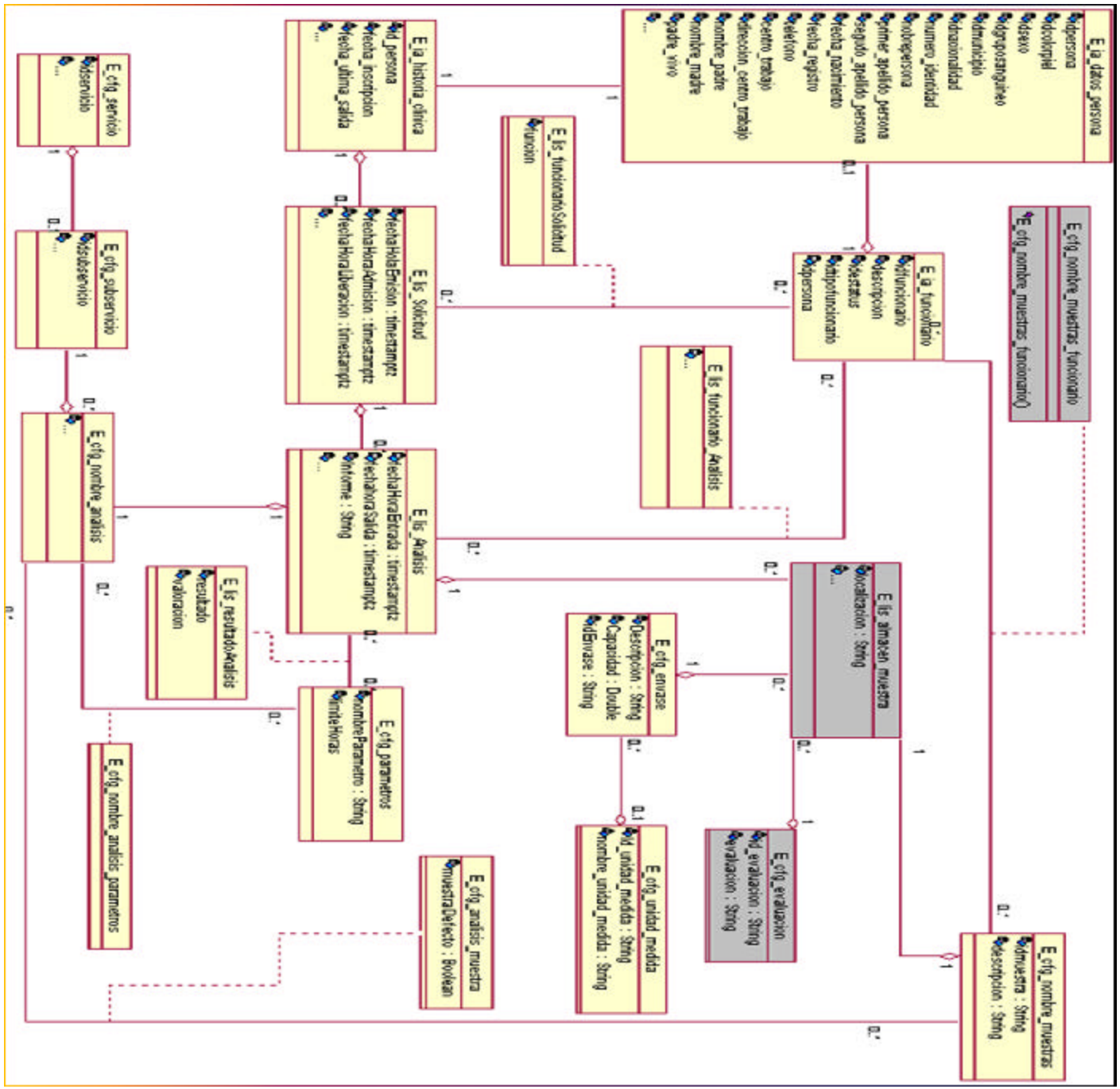
RCU Recepcionar Solicitud (Consultar)



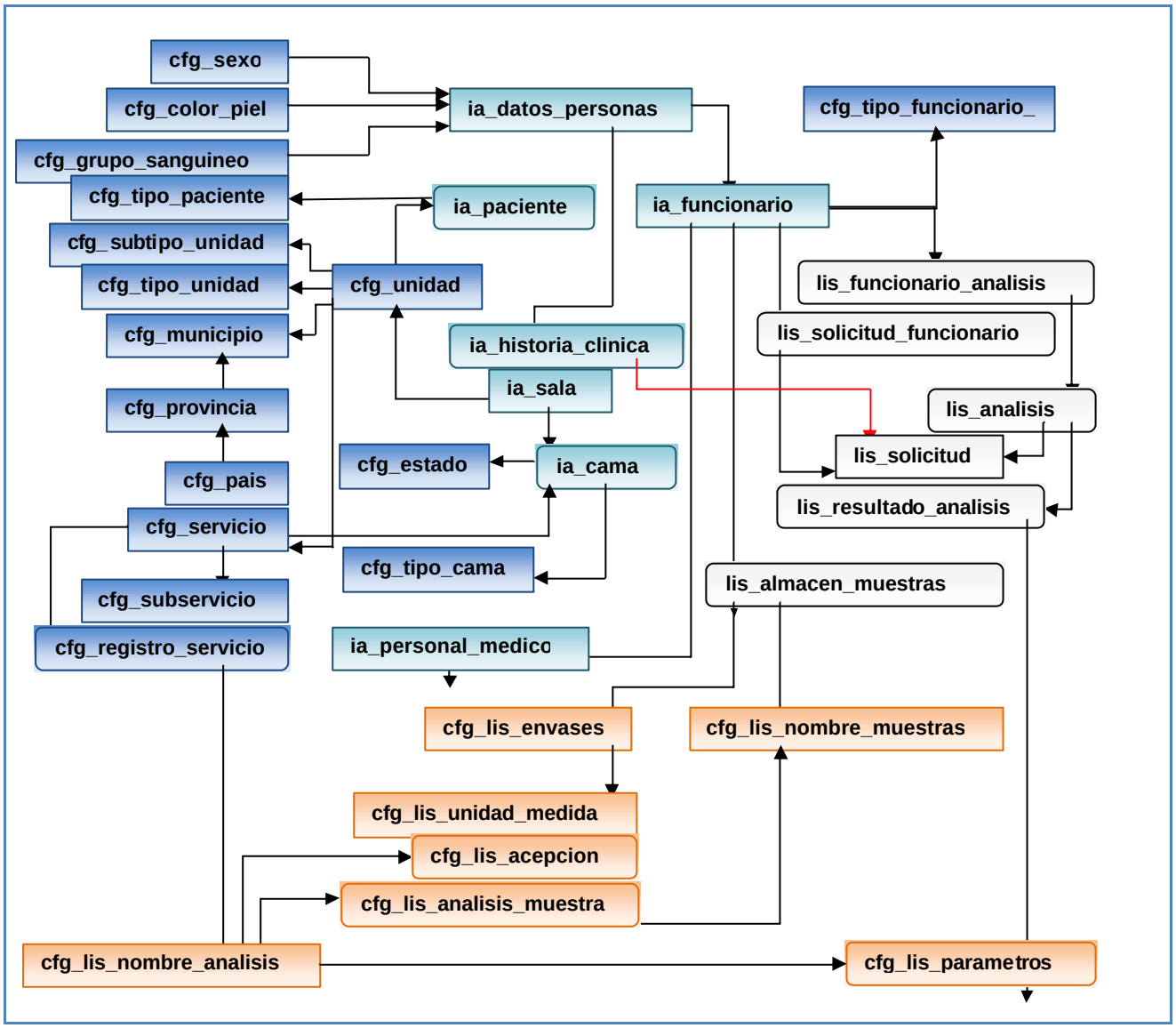
2.2.7 RCU Tomar Muestra



2.2.8 Diseño de la base de datos. Diagrama de clases persistentes.



2.2.9 Modelo de datos. Diagrama entidad-relación



2.3 Análisis de posibles módulos o componentes que son usados.

Este sistema es parte de un sistema mucho mayor (GeHos), por la tanto es necesario hacer uso de otros módulos que como este están en procesos de desarrollo o ya desarrollados.

En este caso, se tiene al de **Configuración**, una especie de módulo auxiliar donde se gestiona toda aquella información común a todo el sistema.

Otro módulo seria el de **Inscripción y admisión**, el cual gestiona todos los procesos que tienen que ver con el paciente y sus datos, este módulo es el que proporciona todo el negocio de búsquedas de pacientes y funcionarios de la salud así como los datos de cada uno de ellos.

Seguridad, módulo que como bien se llama gestiona toda la seguridad y autenticidad de la información de cada unos de los restantes incluyendo LIS.

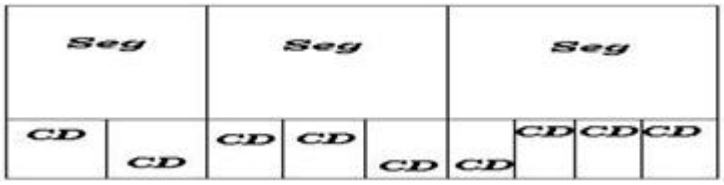
Se hace uso de **NpgSQL,** una librería que proporciona las herramientas para el manejo de datos entre el lenguaje C# y el gestor de bases de datos PostgreSQL estableciendo una comunicación entre la plataforma .NET y el gestor de datos.

El sistema también espera lograr el intercambio de datos, a través del **HL7 (Health Level 7)**, este es un estándar de intercambio de datos electrónicos en el ambiente de la atención médica en la salud, con un gran énfasis en las comunicaciones dentro de los hospitales.

El HL7 actualmente se ocupa de las interfaces entre sistemas que emiten o reciben mensajes de registro, admisión, transferencia y alta de pacientes, pedidos de información al sistema, ordenes, resultados, observaciones clínicas, facturación, reportes y actualización de información de archivos maestros. El mismo no asume ninguna arquitectura en general con respecto a la ubicación de los datos dentro de la aplicación; aunque si está diseñado para dar soporte tanto a un sistema central de atención de pacientes, como a un ambiente más distribuido donde las aplicaciones departamentales son lo repositorios de los datos.

El formato general de los mensajes consiste de campos de datos de longitud variable, separados por caracteres especiales, según reglas específicas de codificación. Los campos de datos se combinan para

formar agrupamientos lógicos denominados segmentos, los cuales a su vez están separados entre sí por caracteres específicos. Un mensaje lo componen diversas líneas cada una de las cuales representan líneas. Atento a lo expuesto, un mensaje del HL7 podría ilustrarse de la siguiente manera:



2.4 Principios de Diseño

2.3.1 Patrón de diseño de fabricación.

Para desarrollar la capa de acceso a datos y negocio se decidió utilizar el patrón de diseño de fabricación, se llaman patrones de fabricación a aquellos patrones que involucran algún tipo de factoría o fábrica (**factory**, en inglés) de objetos.

Con este patrón, se hace que cada tabla de la base de datos se traduzca a la hora de trabajar con ella en la capa de acceso a datos en clases, quiere decir que todo el trabajo con los datos es tratado en su totalidad con objetos, por lo que es invisible para la aplicación, de donde provienen los datos, puede que sea desde un Gestor de bases de datos, PostgresSQL, o SQLServer o MySQL, o si llegan desde un servicio web proveniente de otro sistema o aplicación.

2.3.2 Principios del diseño universal

La aplicación que se ha analizado y diseñado será utilizada por usuarios que tienen en muchos casos conocimientos reducidos de Informática. A partir de lo anteriormente planteado, la aplicación deberá cumplir con los 7 Principios del Diseño Universal o Diseño para Todos:

Principio uso equiparable:
El diseño es útil y vendible a personas con diversas capacidades.

- ✓ Que proporcione las mismas maneras de uso para todos los usuarios: idénticas cuando es posible, equivalentes cuando no lo es.

- ✓ Que evite segregar o estigmatizar a cualquier usuario.
- ✓ Las características de privacidad, garantía y seguridad deben estar igualmente disponibles para todos los usuarios.
- ✓ Que el diseño sea atractivo para todos los usuarios.

Principio uso flexible:

El diseño se acomoda a un amplio rango de preferencias y habilidades individuales.

- ✓ Que ofrezca posibilidades de elección en los métodos de uso.
- ✓ Que pueda accederse y usarse tanto con la mano derecha como con la izquierda.
- ✓ Que facilite al usuario la exactitud y precisión.
- ✓ Que se adapte al paso o ritmo del usuario.

Principio simple e intuitivo:

El uso del diseño es fácil de entender, atendiendo a la experiencia, conocimientos, habilidades lingüísticas o grado de concentración actual del usuario.

- ✓ Que elimine la complejidad innecesaria.
- ✓ Que sea consistente con las expectativas e intuición del usuario.
- ✓ Que se acomode a un amplio rango de alfabetización y habilidades lingüísticas.
- ✓ Que dispense la información de manera consistente con su importancia.
- ✓ Que proporcione avisos eficaces y métodos de respuesta durante y tras la finalización de la tarea.

Principio Información perceptible:

El diseño comunica de manera eficaz la información necesaria para el usuario, atendiendo a las condiciones ambientales o a las capacidades sensoriales del usuario.

- ✓ Que use diferentes modos para presentar de manera redundante la información esencial (gráfica, verbal o táctilmente)
- ✓ Que proporcione contraste suficiente entre la información esencial y sus alrededores.
- ✓ Que amplíe la legibilidad de la información esencial.
- ✓ Que diferencie los elementos en formas que puedan ser descritas (por ejemplo, que haga fácil dar instrucciones o direcciones).

- ✓ Que proporcione compatibilidad con varias técnicas o dispositivos usados por personas con limitaciones sensoriales.

Principio con tolerancia al error:

El diseño minimiza los riesgos y las consecuencias adversas de acciones involuntarias o accidentales.

- ✓ Que disponga los elementos para minimizar los riesgos y errores: elementos más usados, más accesibles; y los elementos peligrosos eliminados, aislados o tapados.
- ✓ Que proporcione advertencias sobre peligros y errores.
- ✓ Que proporcione características seguras de interrupción.
- ✓ Que desaliente acciones inconscientes en tareas que requieren vigilancia.

Principio que exija poco esfuerzo físico:

El diseño puede ser usado eficaz y confortablemente y con un mínimo de fatiga.

- ✓ Que permita que el usuario mantenga una posición corporal neutra.
- ✓ Que utilice de manera razonable las fuerzas necesarias para operar.
- ✓ Que minimice las acciones repetitivas.
- ✓ Que minimice el esfuerzo físico continuado.

Principio tamaño y espacio para el acceso y uso:

Que proporcione un tamaño y espacio apropiados para el acceso, alcance, manipulación y uso, atendiendo al tamaño del cuerpo, la postura o la movilidad del usuario.

- ✓ Que proporcione una línea de visión clara hacia los elementos importantes tanto para un usuario sentado como de pie.
- ✓ Que el alcance de cualquier componente sea confortable para cualquier usuario sentado o de pie.
- ✓ Que se acomode a variaciones de tamaño de la mano o del agarre.
- ✓ Que proporcione el espacio necesario para el uso de ayudas técnicas o de asistencia personal.

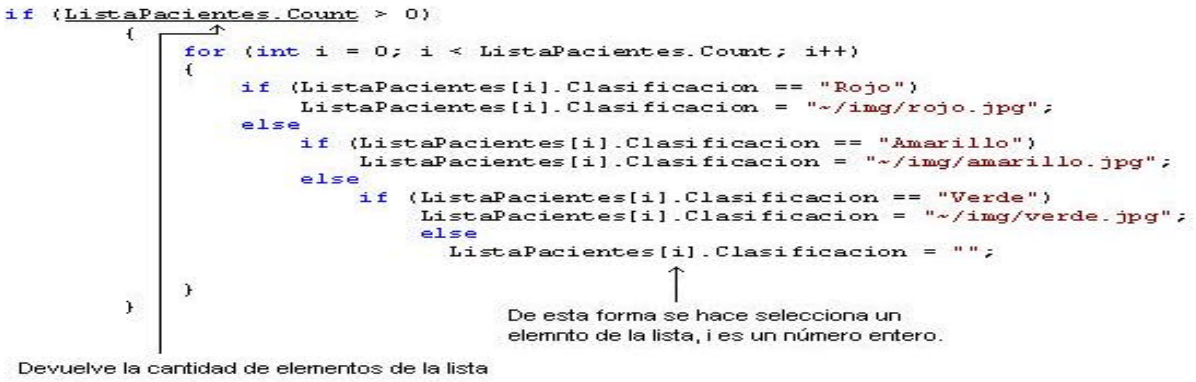
2.4 Estructuras de datos utilizadas

Para el trabajo con listas en .Net con el lenguaje C#, hay que incluir una librería en la que está incluida todas estas operaciones y aparte una serie de funciones que permiten el trabajo con las mismas. Se incluye System.Collections.Generic, ya posteriormente se puede trabajar con List<*tipo*>, donde se le especifica de que tipo de datos se quiere crear la lista.

Ejemplo:

Para mostrar el listado de recepciones que se han hecho en el día hace falta tenerlas en una lista, entonces se declara:

Luego se necesita asignarle a cada elemento de la lista una clasificación, para hacer esto se necesito recorrer la lista de pacientes y para ello se hizo uso de una función que tiene la lista, es "Count" la cual devuelve la cantidad de elementos de esa lista. Vea la siguiente imagen que lo que muestra:



En sentido general el uso de las listas facilitó mucho más el trabajo en la implementación, aunque también hay que reconocer el carácter eficiente que tienen las mismas, por la forma en que están organizadas.

2.5 Descripción de las principales clases usadas

En esta sección se hará una descripción de aquellas clases usadas en el desarrollo de la aplicación, las entidades, las controladoras y las clases interfaz.

2.5.1 Clases entidad

Nombre: Acepción	
Tipo de clase: Entidad	
Atributo	**Tipo**
idAcepcion	Int32
idNombreAnalisis	String
Descripción	String
defecto	Boolean
abreviatura	String
Para cada responsabilidad:	
Nombre:	Acepcion(Int32 idAcepcion, String idNombreAnalisis, String Descripción, Boolean defecto, String abreviatura)
Descripción:	Constructor de la clase

Nombre: Analisis	
Tipo de clase: Entidad	
Atributo	**Tipo**
idAnalisis	String
idNombreAnalisis	String
idSolicitud	String
fechaEntrada	DateTime
fechaSalida	DateTime
informe	String

notaEspecialTecnico	String
idEstadoExamen	Int32
Para cada responsabilidad:	
Nombre:	Analisis(String idAnalisis, String idNombreAnalisis, String idSolicitud, DateTime fechaEntrada, DateTime fechaSalida, String informe, String notaEspecialTecnico, Int32 idEstadoExamen)
Descripción:	Constructor de la clase

Nombre: AnalisisMuestra	
Tipo de clase: Entidad	
Atributo	**Tipo**
idNombreMuestra	Int32
idNombreAnalisis	String
muestraDefecto	Boolean
Para cada responsabilidad:	
Nombre:	AnalisisMuestra(Int32 idNombreMuestra, String idNombreAnalisis, Boolean muestraDefecto)
Descripción:	Constructor de la clase

Nombre: DatoAlmacenaje	
Tipo de clase: Entidad	
Atributo	**Tipo**
idDatoAlmacenaje	Int32
idTemperatura	Int32
idMuestra	String
codigoProducto	String
datoAdicional	String
fechaHoraInicial	DateTime
fechaHoraSalidaEstimada	DateTime
fechaSalidaReal	DateTime
motivoEstadia	String

fotosencible	Boolean
Para cada responsabilidad:	
Nombre:	DatoAlmacenaje(Int32 idDatoAlmacenaje, Int32 idTemperatura, String idMuestra, String codigoProducto, String datoAdicional, DateTime fechaHoraInicial, DateTime fechaHoraSalidaEstimada, DateTime fechaSalidaReal, String motivoEstadia, Boolean fotosencible)
Descripción:	Constructor de la clase

Nombre: DatoMuestra	
Tipo de clase: Entidad	
Atributo	**Tipo**
idMuestra	String
idNombreMuestra	Int32
idAnalisis	String
idFuncionario	Int32
fechaHoraRecepcion	DateTime
localizacion	String
notaEspecialMuestra	String
Para cada responsabilidad:	
Nombre:	DatoMuestra(String idMuestra, Int32 idNombreMuestra, String idAnalisis, Int32 idFuncionario, DateTime fechaHoraRecepcion, String localizacion, String notaEspecialMuestra)
Descripción:	Constructor de la clase

Nombre: Envases	
Tipo de clase: Entidad	
Atributo	**Tipo**
idEnvase	String
idUnidadMedida	String
Descripción	String
capacidad	Double

Para cada responsabilidad:	
Nombre:	Envases(String idEnvase, String idUnidadMedida, String Descripción, Double capacidad)
Descripción:	Constructor de la clase

Nombre: EstadoExamen	
Tipo de clase: Entidad	
Atributo	**Tipo**
idEstadoExamen	Int32
Descripción	String
Para cada responsabilidad:	
Nombre:	EstadoExamen(Int32 idEstadoExamen, String Descripción)
Descripción:	Constructor de la clase

Nombre: EvaluacionMuestra	
Tipo de clase: Entidad	
Atributo	**Tipo**
idEvaluacion	String
Descripción	String
Para cada responsabilidad:	
Nombre:	EvaluacionMuestra(String idEvaluacion, String Descripción)
Descripción:	Constructor de la clase

Nombre: NombreEstado	
Tipo de clase: Entidad	
Atributo	**Tipo**
idNombreEstado	String
Descripción	String
Para cada responsabilidad:	
Nombre:	NombreEstado(String idNombreEstado, String Descripción)
Descripción:	Constructor de la clase

Nombre: NombreMuestra **Tipo de clase:** Entidad	
Atributo	**Tipo**
idNombreMuestra	Int32
Descripción	String
Para cada responsabilidad:	
Nombre:	NombreMuestra(Int32 idNombreMuestra, String Descripción)
Descripción:	Constructor de la clase

Nombre: Parametro	
Tipo de clase: Entidad	
Atributo	**Tipo**
idParametro	String
nombre tiempoLimiteHoras	String Int32
tipoParametro	String
Para cada responsabilidad:	
Nombre:	Parametro(String idParametro, String nombre, Int32 tiempoLimiteHoras, String tipoParametro)
Descripción:	Constructor de la clase

Nombre: RazonEstado	
Tipo de clase: Entidad	
Atributo	**Tipo**
idAnalisis	String
Descripción	String
Para cada responsabilidad:	
Nombre:	RazonEstado(String idAnalisis, String Descripción)
Descripción:	Constructor de la clase

Nombre: ResultadosAnalisis **Tipo de clase:** Entidad	
Atributo	**Tipo**
idAnalisis	String
idParametro	String
resultado	String
Para cada responsabilidad:	
Nombre:	ResultadosAnalisis(String idAnalisis, String idParametro, String resultado)
Descripción:	Constructor de la clase

Nombre: SolicitudAnalisis	
Tipo de clase: Entidad	
Atributo	**Tipo**
idSolicitud idPrioridad	String Int32
idPersona	Int32
fechaAdmision	DateTime
fechaLiberacion	DateTime
recomendacionesPrev	String
fechaEmision	DateTime
Para cada responsabilidad:	
Nombre:	SolicitudAnalisis(String idSolicitud, Int32 idPrioridad, Int32 idPersona, DateTime fechaAdmision, DateTime fechaLiberacion, String recomendacionesPrev, DateTime fechaEmision)
Descripción:	Constructor de la clase

Nombre: Temperatura	
Tipo de clase: Entidad	
Atributo	**Tipo**
idTemperatura	Int32
idUnidadMedida	String

tempMin	Int32
tempMax	Int32
Para cada responsabilidad:	
Nombre:	Temperatura(Int32 idTemperatura, String idUnidadMedida, Int32 tempMin, Int32 tempMax)
Descripción:	Constructor de la clase

Nombre: TipoFuncionario	
Tipo de clase: Entidad	
Atributo	**Tipo**
idTipoFuncionario	Int32
Descripción	String
Para cada responsabilidad:	
Nombre:	TipoFuncionario(Int32 idTipoFuncionario, String Descripcion)
Descripción:	Constructor de la clase

Nombre: UnidadMedida	
Tipo de clase: Entidad	
Atributo	**Tipo**
idUnidadMedida	String
unidad	String
Para cada responsabilidad:	
Nombre:	UnidadMedida(String idUnidadMedida, String unidad)
Descripción:	Constructor de la clase

Nombre: UsoEnvase	
Tipo de clase: Entidad	
Atributo	**Tipo**
idUsoEnvase	Int32
idEnvase	String
Descripción	String
Para cada responsabilidad:	

Nombre:	UsoEnvase(Int32 idUsoEnvase, String idEnvase, String Descripción)
Descripción:	Constructor de la clase

2.5.2 Clases Controladoras

Nombre: GestionarSolicitud	
Tipo de clase: Controladora	
Atributo	**Tipo**
solicitud	SolicitudAnalisis
solicitudes	List<SolicitudAnalisis>
Para cada responsabilidad:	
Nombre:	GestionarSolicitud()
Descripción:	Constructor de la clase
Nombre:	BuscarSolicitud()
Descripción:	Método en el cual se busca una Solicitud, tiene varias sobrecargas, las cuales especifican los parámetros de búsqueda de una solicitud que se solicite. Salida: SolicitudAnalisis
Nombre:	ActualizarSolicitud(SolicitudAnalisis solicitud)
Descripción:	Método en el cual se actualiza una solicitud.
Nombre:	AdicionarSolicitud(SolicitudAnalsis solicitud)
Descripción:	Método en el cual se adiciona una solicitud nueva.
Nombre:	EliminarSolicitud(string idSolicitud)
Descripción:	Método en el cual se elimina una solicitud según su id.

Nombre: GestionarAnalisis	
Tipo de clase: Controladora	
Atributo	**Tipo**
analisis listAnalisis	Analisis List<Analisis>
Para cada responsabilidad:	
Nombre:	GestionarAnalisis()

Descripción:	Constructor de la clase
Nombre:	BuscarUnAnalisis()
Descripción:	Método en el cual se busca un análisis, tiene varias sobrecargas, las cuales especifican los parámetros de búsqueda de un análisis que se solicite. Salida: Analisis
Nombre:	BuscarAnalisis()
Descripción:	Método en el cual se busca una lista de analisis, tiene varias sobrecargas, las cuales especifican los parámetros de búsqueda. Salida: List<Analisis>
Nombre:	ActualizarAnalisis(Analisis analisis)
Descripción:	Método en el cual se actualiza un análisis.
Nombre:	AdicionarAnalisis(Analisis analisis)
Descripción:	Método en el cual se adiciona un análisis.
Nombre:	EliminarAnalisis(string idAnalisis)
Descripción:	Método en el cual se elimina un análisis según su id.

Nombre: GestionarMuestra	
Tipo de clase: Controladora	
Atributo	**Tipo**
muestra	DatoMuestra
listMuestras	List<DatoMuestra>
Para cada responsabilidad:	
Nombre:	GestionarMuestra()
Descripción:	Constructor de la clase
Nombre:	BuscarUnMuestra()
Descripción:	Método en el cual se busca una muestra, tiene varias sobrecargas, las cuales especifican los parámetros de búsqueda de una muestra que se solicite. Salida: DatoMuestra
Nombre:	BuscarMuestras()
Descripción:	Método en el cual se busca una lista de muestras, tiene varias sobrecargas, las cuales especifican los parámetros de búsqueda. Salida: List<DatoMuestra>

Nombre: Descripción:	ActualizarMuestra(DatoMuestra muestra) Método en el cual se actualiza una muestra.
Nombre:	AdicionarMuestra(DatoMuestra muestra)
Descripción:	Método en el cual se adiciona una muestra
Nombre:	EliminarMuestra(string idMuestra)
Descripción:	Método en el cual se elimina una muestra según su id.

2.5.3 Clases Interfaz

Nombre: EmitirSolicitud	
Tipo de clase: Interfaz	
Atributo	**Tipo**
solicitud	SolicitudAnalisis
datosPersonas	DatosPersonas
listDatosPersonas	List<DatosPersonas>
datosPersonaDoctor	DatosPersonas
listDatosDoctor	List<DatosPersonas>
Ademas un grupo de controles web usados en los formularios que se usan en la página	Controles Web
Para cada responsabilidad:	
Nombre:	Page_Load(object sender, EventArgs e)
Descripción:	Función del evento onload de la página
Nombre:	chblistTiposEx_SelectedIndexChanged(object sender, EventArgs e)
Descripción:	Función del evento SelectedIndexChanged del ChexboxList TiposEx
Nombre:	btnExSel_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button btnExSel
Nombre:	btnExUnSel_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button btnExUnSel
Nombre:	lsbExamenesSel_SelectedIndexChanged(object sender, EventArgs e)
Descripción:	Función del evento SelectedIndexChanged del ListBox ExamenesSel
Nombre:	lsbExamenes_SelectedIndexChanged(object sender, EventArgs e)
Descripción:	Función del evento SelectedIndexChanged del ListBox Examenes
Nombre:	lsbTipoMuestra_SelectedIndexChanged(object sender, EventArgs e)

Descripción:	Función del evento SelectedIndexChanged del ListBox TipoMuestra
Nombre:	chbInstEspeciales_CheckedChanged(object sender, EventArgs e)
Descripción:	Función del evento SelectedIndexChanged del ChekBoxList InstEspeciales
Nombre:	btnEmitirAceptar_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button EmitirAceptar
Nombre:	buttonAceptar_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button Aceptar
Nombre:	btnBuscarDoctor_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button BuscarDoctor
Nombre:	btnAceptarDoctor_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button AceptarDoctor

Nombre: BandejaLiberacion	
Tipo de clase: Interfaz	
Atributo	**Tipo**
idSolicitudLiberacion	String
Ademas un grupo de controles web usados en los formularios que se usan en la página	Controles Web
Para cada responsabilidad:	
Nombre:	Page_Load(object sender, EventArgs e)
Descripción:	Función del evento onload de la página
Nombre:	gdwLiberacion_SelectedIndexChanged1(object sender, EventArgs e)
Descripción:	Función del evento SelectedIndexChanged del GridView gdwLiberacion

Nombre: BandejaMuestras	
Tipo de clase: Interfaz	
Atributo	**Tipo**
idSolicitudMuestras	string
Ademas un grupo de controles web usados en los formularios que se usan en la página	Controles Web
Para cada responsabilidad:	
Nombre:	Page_Load(object sender, EventArgs e)

Descripción:	Función del evento onload de la página
Nombre:	gdwRecepcionMuestras_SelectedIndexChanged(object sender, EventArgs e)
Descripción:	Función del evento SelectedIndexChanged del GridView gdwRecepcionMuestras

Nombre: BandejaRecepcion	
Tipo de clase : Interfaz	
Atributo	**Tipo**
idSolicitudSelecionada	String
Ademas un grupo de controles web usados en los formularios que se usan en la página	Controles Web
Para cada responsabilidad:	
Nombre:	Page_Load(object sender, EventArgs e)
Descripción:	Función del evento onload de la página
Nombre:	gdwBRecepcion_SelectedIndexChanged(object sender, EventArgs e)
Descripción:	Función del evento SelectedIndexChanged del GridView gdwBRecepcion

Nombre: BandejaTecnico	
Tipo de clase : Interfaz	
Atributo	**Tipo**
idAnalisisSeleccionado	String
listA	List<Analisis>
Ademas un grupo de controles web usados en los formularios que se usan en la página	Controles Web
Para cada responsabilidad:	
Nombre:	Page_Load(object sender, EventArgs e)
Descripción:	Función del evento onload de la página

Nombre: ConsultarSolicitud	
Tipo de clase : Interfaz	
Atributo	**Tipo**
idSolicitudSelecionada	String
placeTecnicos	PlaceHolder

Además un grupo de controles web usados en los formularios que se usan en la página	Controles Web
Para cada responsabilidad:	
Nombre:	Page_Load(object sender, EventArgs e)
Descripción:	Función del evento onload de la página
Nombre:	btnAceptar_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button btnAceptar

Nombre: EmitirResultado	
Tipo de clase: Interfaz	
Atributo	**Tipo**
idAnalisisSeleccionado	String
tabla	Table
datos	DatosAnalsiis_Resultados
Además un grupo de controles web usados en los formularios que se usan en la página	Controles Web
Para cada responsabilidad:	
Nombre:	Page_Load(object sender, EventArgs e)
Descripción:	Función del evento onload de la página

Nombre: LiberarAnalsiis	
Tipo de clase: Interfaz	
Atributo	**Tipo**
idAnalisisLiberar	String
analisis	Analisis
muestra	DatoMuestra
acepcion	Acepcion
Además un grupo de controles web usados en los formularios que se usan en la página	Controles Web
Para cada responsabilidad:	
Nombre:	Page_Load(object sender, EventArgs e)
Descripción:	Función del evento onload de la página
Nombre:	btnAceptarL_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button btnAceptarL

Nombre: Descripción:	rdblLiberar_SelectedIndexChanged(object sender, EventArgs e) Función del evento SelectedIndexChanged del RadioButtonList rdblLiberar
Nombre:	btnAceptarNV_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button btnAceptarNV
Nombre:	btnAceptarNV_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button btnAceptarNV

Nombre: SolicitudLiberar	
Tipo de clase: Interfaz	
Atributo	**Tipo**
idSolicitudLiberacion	String
idAnalisisLiberar	String
listExamenesGdwLiberar Además un grupo de controles web usados en los formularios que se usan en la página	List<ExamenMuestra> Controles Web
Para cada responsabilidad:	
Nombre:	Page_Load(object sender, EventArgs e)
Descripción:	Función del evento onload de la página
Nombre:	btnTomarMuestra_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button btnTomarMuestra

Nombre: TomarMuestra	
Tipo de clase: Interfaz	
Atributo	**Tipo**
idSolicitudMuestras	String
listExamenesGdw	List<ExamenMuestra>
listT Además un grupo de controles web usados en los formularios que se usan en la página	List<RangoTemperatura> Controles Web
Para cada responsabilidad:	
Nombre:	Page_Load(object sender, EventArgs e)
Descripción:	Función del evento onload de la página
Nombre:	ddlMovMuestra_SelectedIndexChanged(object sender, EventArgs e)

Descripción:	Función del evento SelectedIndexChanged del DropDownListMovMuestra
Nombre:	chbReactivo_CheckedChanged(object sender, EventArgs e)
Descripción:	Función del evento CheckedChanged del CheckBox chbReactivo
Nombre:	chbEstMuestra_CheckedChanged(object sender, EventArgs e)
Descripción:	Función del evento CheckedChanged del CheckBox chbEstMuestra
Nombre:	chbComentario_CheckedChanged(object sender, EventArgs e)
Descripción:	Función del evento CheckedChanged del CheckBox chbComentario
Nombre:	btnTomarMuestra_Click(object sender, EventArgs e)
Descripción:	Función del evento CheckedChanged del Button btnTomarMuestra
Nombre:	gdwlistaMuestras_SelectedIndexChanged(object sender, EventArgs e)
Descripción:	Función del evento SelectedIndexChanged del GridView gdwListMuestras
Nombre:	btnTomaTerminada_Click(object sender, EventArgs e)
Descripción:	Función del evento onClick del Button btnTomaTerminada

2.6 Formatos de reportes

Los reportes se obtendrán en tablas que en algunos casos pueden tener una gran cantidad de elementos en dependencia de la información a visualizar, por lo que debe quedar pautado que hasta un total de 25 resultados la tabla funcione con scroll, y para más de esta cantidad será entonces por paginado, organizado por números consecutivos, con enlaces a los resultados restantes, exceptuando el activo en ese preciso momento.

En algunos casos se hará uso de pequeñas imágenes que indicarán funcionalidades surgidas a partir de la visualización de estos reportes.

Los reportes serán concebidos sobre ventanas, utilizando un formato de letra clara, legible y con colores claros para no recargar y hacer engorrosa su impresión.

2.7 Diseño de Interfaz Gráfica del Proyecto GeHos

Para lograr una mayor eficiencia en el proceso de trabajo, y sobre todo para lograr una coherencia formal entre todos los módulos del sistema, y que sean identificados así como parte de un todo, se han pautado una serie de elementos comunes que facilitarán su reconocimiento y el uso que se haga de ellos.

Se diseñará una Pantalla Inicial global del **GeHos,** desde la cual se accederá a los diferentes módulos. Esta pantalla contará con accesos a los diferentes módulos, informaciones generales, guías de ayuda, sistema de avisos que genera cada registro y enlaces definidos.

Así mismo será diseñada una Pantalla Inicial para cada una de las aplicaciones, que contará con accesos a todas las utilidades, avisos, ayuda y un enlace para regresar a la pantalla Inicial del **GeHos**.

Su diseño está determinado fundamentalmente por el principio de la usabilidad, teniendo en cuenta que no se trata de un sitio web, sino de una aplicación de trabajo donde el diseño tiene como principal propósito facilitar su uso, comprensión y navegación, por encima de ornamentos inútiles, aunque manteniendo pautas estéticas, orgánicas y agradables. Formalmente, usabilidad se define como la medida en que un producto puede ser usado por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción, en un contexto de uso especificado [ISO 9241-11].

2.8 Concepción general de la ayuda

Teniendo en cuenta que las ayudas en las aplicaciones Web en general no suelen ser explicaciones detallistas del sistema informático al cual representan, sino que son generalmente simples aclaraciones, informaciones generales de la aplicación o datos de la empresa que le da soporte o realizó el producto se está proponiendo que la ayuda estará accesible como parte del menú en todas las páginas de la aplicación, con el objetivo de que el usuario vea la información que necesita en ese momento. Cada página mostrará como realizar aquellas operaciones que estén relacionadas con la posición donde se encuentre el usuario en dicho momento, además se aportan los conceptos que se manejan en la aplicación, para que el usuario se familiarice con algunas entradas, el entorno de la elaboración de los informes o reportes y otras funcionalidades que se le brindan en el sistema.

La ayuda para todas las aplicaciones del Sistema Integral de Salud estará concebida bajo los principios del soporte técnico en línea, que es una práctica muy utilizada en las aplicaciones Web dinámicas, como la que se está desarrollando y generalmente se implementará con una explicación general de las opciones y con vínculos a sistemas de correo o a otros sitios Web.

También se podrá contar, con un soporte técnico en línea fuera de la aplicación principal para que los usuarios puedan informar acerca de errores que suceden en la aplicación, emitir sugerencias de su funcionalidad o recibir soluciones a las preguntas que de forma "directa" pueden realizar a los administradores y creadores del producto.

Esta forma de ayuda resulta de gran ventaja, ya que contribuye a la resolución de problemas en el software, la gestión de cambios y configuraciones y la actualización y el mantenimiento del producto.

Además, se tendrá en cuenta la confección de manuales de usuario y será entregado a los usuarios de cada módulo un manual de usuario en formato digital o en papel, que explicará de forma detallada las principales funcionalidades y opciones que brinda el software.

Está concebida una capacitación técnica directa durante la etapa de implantación del producto, ya que por parte de la empresa productora y bajo acuerdo con los clientes se coordinará la puesta en marcha de cursos de capacitación o entrenamiento en el uso de la aplicación, dirigida a todos los usuarios potenciales antes y durante la implantación oficial de este producto en el Sistema Nacional de Salud.

Conclusiones

Luego de seguir, durante el desarrollo de esta investigación, las distintas etapas que propone RUP para el desarrollo de un software, se ha logrado obtener un modelo bien detallado de la aplicación que se desea, y que constituya la solución más asequible, y a partir de este modelo se ha logrado implementar un sistema, el cual cumple con toda la funcionalidad que requiere la situación problémica existente

CAPITULO 3

Validación de la solución propuesta

Introducción

Cuando aparecieron los primeros grandes sistemas informáticos, se incluyó a nivel metódico un nuevo proceso en la confección de los mismos: el proceso de prueba. Hoy en día se calcula que la fase o proceso de pruebas representa más de la mitad del coste de un programa. Ya que requiere, un tiempo similar al de la programación lo que obviamente acarrea un alto costo económico, cuando estos no involucran vidas humanas, puesto que en este último caso el costo suele superar el 80% siendo esta etapa más cara que el propio desarrollo y diseño de los distintos programas que conforman el sistema.

Hacer las prueba infalibles de un programa, implicaría ponerlo en todas las situaciones posibles, de esta manera se aseguraría que el mismo se encuentra completamente libre de errores, como se imaginarán esto es imposible porque a pesar de que el número de líneas que lo conforman es finito, la prueba pasa a ser infinita cuando entran en juego los bucles con lo que hacer la prueba empírica exacta pasa de ser una enorme e incalculable cantidad de posibilidades a una cifra ciertamente infinita. Considerando este último punto lo que queda por hacer es buscar formas y métodos abordables para acercarse lo más posible a un resultado óptimo.

3.1 Fases de pruebas

Hay multitud de conceptos (y palabras clave) asociadas a las tareas de prueba. Clasificarlas es difícil, pues no son mutuamente disjuntas, sino muy entrelazadas. En lo que sigue se intentará la siguiente estructura para la presentación:

Fases de prueba:

- UNIDADES
  Planteamientos:
  - CAJA BLANCA
    Cobertura:
    - de segmentos
    - de ramas

- de condición/decisión
- de bucles

- CAJA NEGRA

Cobertura de requisitos

- INTEGRACIÓN
- ACEPTACIÓN

La prueba de unidades se plantea a pequeña escala, y consiste en ir probando uno a uno los diferentes módulos que constituyen una aplicación.

Las pruebas de integración y de aceptación son pruebas a mayor escala, que puede llegar a dimensiones industriales cuando el número de módulos es muy elevado, o la funcionalidad que se espera del programa es muy compleja.

Las pruebas de integración se centran en probar la coherencia semántica entre los diferentes módulos, tanto de semántica estática (se importan los módulos adecuados; se llama correctamente a los procedimientos proporcionados por cada módulo), como de semántica dinámica (un módulo recibe de otro lo que esperaba). Normalmente estas pruebas se van realizando por etapas, englobando progresivamente más y más módulos en cada prueba.

Por último, las pruebas de aceptación son las que se plantea el cliente final, que decide qué pruebas va a aplicarle al producto antes de darlo por bueno y pagarlo. De nuevo, el objetivo del que prueba es encontrar los fallos lo antes posible, en todo caso antes de pagarlo y antes de poner el programa en producción.

Pero no se utilizara ni pruebas de integración ya que estas son para probar un número grande módulos, ni las pruebas de aceptación ya que el producto no tiene un cliente como tal que exija hacerle pruebas. En este trabajo solo se han realizado las pruebas de unidades.

3.2 Prueba de unidades

¿Cómo se prueban módulos sueltos?

Normalmente cabe distinguir una fase informal antes de entrar en la fase de pruebas propiamente dicha. La fase informal la lleva a cabo el propio codificador en su despacho, y consiste en ir ejecutando el código para convencerse de que "básicamente, funciona". Esta fase suele consistir en pequeños ejemplos que se

intentan ejecutar. Si el módulo falla, se suele utilizar un depurador para observar la evolución dinámica del sistema, localizar el fallo, y repararlo.

En lenguajes antiguos, poco rigurosos en la sintaxis y/o en la semántica de los programas, esta fase informal llega a ser muy dura, laboriosa, y susceptible de dejar pasar grandes errores sin que se note. En lenguajes modernos, con reglas estrictas, hay herramientas que permiten análisis exhaustivos de los aspectos estáticos de la semántica de los programas: tipado de las variables, ámbitos de visibilidad, parámetros de llamada a procedimientos, etc.

Hay asimismo, herramientas más sofisticadas capaces de emitir "opiniones" sobre un programa y alertar de construcciones arriesgadas, de expresiones muy complicadas (que se prestan a equivocaciones), etc. A veces, pueden prevenir sobre variables que pueden usarse antes de tomar algún valor (no inicializadas), variables que se cargan pero luego no se usan, y otras posibilidades que, sin ser necesariamente errores en sí mismas, sí suelen apuntar a errores de verdad.

Más adelante, cuando el módulo parece presentable, se entra en una fase de prueba sistemática. En esta etapa se empieza a buscar fallos siguiendo algún criterio para que "no se escape nada". Los criterios más habituales son los denominados de caja negra y de caja blanca.

Se dice que una prueba es de caja negra cuando prescinde de los detalles del código y se limita a lo que se ve desde el exterior. Intenta descubrir casos y circunstancias en los que el módulo no hace lo que se espera de él.

Por oposición al término "caja negra" se suele denominar "caja blanca" al caso contrario, es decir, cuando lo que se mira con lupa es el código que está ahí escrito y se intenta que falle. Quizás sea más propia la denominación de "pruebas de caja transparente".

3.2.1 Pruebas de Caja Blanca

Sinónimos:

- pruebas estructurales
- pruebas de caja transparente

En estas pruebas se está siempre observando el código, que las pruebas se dedican a ejecutar con ánimo de "probarlo todo". Esta noción de prueba total se formaliza en lo que se llama "cobertura" y no es sino una medida porcentual de ¿cuánto código se ha cubierto?

Hay diferentes posibilidades de definir la cobertura. Todas ellas intentan sobrevivir al hecho de que el número posible de ejecuciones de cualquier programa no trivial es (a todos los efectos prácticos) infinito. Pero si el 100% de cobertura es infinito, ningún conjunto real de pruebas pasaría de un infinitésimo de cobertura. Esto puede ser muy interesante para los matemáticos; pero no sirve para nada.

Cobertura de segmentos

A veces también denominada "cobertura de sentencias". Por segmento se entiende una secuencia de sentencias sin puntos de decisión. Como el ordenador está obligado a ejecutarlas una tras otra, es lo mismo decir que se han ejecutado todas las sentencias o todos los segmentos.

El número de sentencias de un programa es finito. Basta coger el código fuente e ir contando. Se puede diseñar un plan de pruebas que vaya ejercitando más y más sentencias, hasta que se haya pasado por todas, o por una inmensa mayoría.

En la práctica, el proceso de pruebas termina antes de llegar al 100%, pues puede ser excesivamente laborioso y costoso provocar el paso por todas y cada una de las sentencias.

A la hora de decidir el punto de corte antes de llegar al 100% de cobertura hay que ser precavido y tomar en consideración algo más que el índice conseguido. En efecto, ocurre con harta frecuencia que los programas contienen código muerto o inalcanzable. Puede ser que este trozo del programa, simplemente "sobre" y se pueda prescindir de él; pero a veces significa que una cierta funcionalidad, necesaria, es inalcanzable: esto es un error y hay que corregirlo.

Cobertura de ramas

La cobertura de segmentos es engañosa en presencia de segmentos opcionales. Por ejemplo:

IF Condición THEN EjecutaEsto; END;

Desde el punto de vista de cobertura de segmentos, basta ejecutar una vez, con éxito en la condición, para cubrir todas las sentencias posibles. Sin embargo, desde el punto de vista de la lógica del programa, también debe ser importante el caso de que la condición falle (si no lo fuera, sobra el IF). Sin embargo, como en la rama ELSE no hay sentencias, con 0 ejecuciones se tiene el 100%.

Para afrontar estos casos, se plantea un refinamiento de la cobertura de segmentos consistente en recorrer todas las posibles salidas de los puntos de decisión. Para el ejemplo de arriba, para conseguir una cobertura de ramas del 100% hay que ejecutar (al menos) 2 veces, una satisfaciendo la condición, y otra no.

Estos criterios se extienden a las construcciones que suponen elegir 1 de entre varias ramas. Por ejemplo, el CASE.

Nótese que si se lograra una cobertura de ramas del 100%, esto llevaría implícita una cobertura del 100% de los segmentos, pues todo segmento está en alguna rama. Esto es cierto salvo en programas triviales que carecen de condiciones (a cambio, basta 1 sola prueba para cubrirlo desde todos los puntos de vista). El criterio también debe refinarse en lenguajes que admiten excepciones (por ejemplo, Ada). En estos casos, hay que añadir pruebas para provocar la ejecución de todas y cada una de las excepciones que pueden dispararse.

Cobertura de condición/decisión

La cobertura de ramas resulta a su vez engañosa cuando las expresiones booleanas que se usan para decidir por qué rama tirar son complejas. Por ejemplo:

IF Condición1 OR Condición2 THEN HazEsto; END;

Las condiciones 1 y 2 pueden tomar 2 valores cada una, dando lugar a 4 posibles combinaciones. No obstante sólo hay dos posibles ramas y bastan 2 pruebas para cubrirlas. Pero con este criterio se logran otras combinaciones de las condiciones.

Si se considera sobre el caso anterior las siguientes pruebas:

Prueba 1: Condicion1 = TRUE y Condicion2 = FALSE

Prueba 2: Condicion1 = FALSE y Condicion2 = TRUE

Prueba 3: Condicion1 = FALSE y Condicion2 = FALSE

Prueba 4: Condicion1 = TRUE y Condicion2 = TRUE

Bastan las pruebas 2 y 3 para tener cubiertas todas las ramas. Pero con ellos sólo se ha probado una posibilidad para la Condición1.

Para afrontar esta problemática se define un criterio de cobertura de condición/decisión que trocea las expresiones booleanas complejas en sus componentes e intenta cubrir todos los posibles valores de cada uno de ellos.

Nótese que no basta con cubrir cada una de las condiciones componentes, si no que además hay que cuidar de sus posibles combinaciones de forma que se logre siempre probar todas y cada una de las ramas. Así, en el ejemplo anterior no basta con ejecutar las pruebas 1 y 2, pues aun cuando se cubre perfectamente cada posibilidad de cada condición por separado, lo que no se ha logrado es recorrer las dos posibles ramas de la decisión combinada. Para ello es necesario añadir la prueba 3.

El conjunto mínimo de pruebas para cubrir todos los aspectos es el formado por las pruebas 3 y 4. Aún así, nótese que no se ha probado todo lo posible. Por ejemplo, si en el programa se pone AND donde se quería poner OR (o viceversa), este conjunto de pruebas no lo detecta. Sólo se quiere decir que la cobertura es un criterio útil y práctico; pero no es prueba exhaustiva.

Cobertura de bucles

Los bucles no son más que segmentos controlados por decisiones. Así, la cobertura de ramas cubre plenamente la esencia de los bucles. Pero eso es simplemente la teoría, pues la práctica descubre que los bucles son una fuente inagotable de errores, todos triviales, algunos mortales. Un bucle se ejecuta un cierto número de veces; pero ese número de veces debe ser muy preciso, y lo más normal es que ejecutarlo una vez de menos o una vez de más tenga consecuencias indeseables. Y, sin embargo, es extremadamente fácil equivocarse y redactar un bucle que se ejecuta 1 vez de más o de menos.

Para un bucle de tipo WHILE hay que pasar 3 pruebas

1. 0 ejecuciones
2. 1 ejecución
3. más de 1 ejecución

Para un bucle de tipo REPEAT hay que pasar 2 pruebas

1. 1 ejecución
2. más de 1 ejecución

Los bucles FOR, en cambio, son muy seguros, pues en su cabecera está definido el número de veces que se va a ejecutar. Ni una más, ni una menos, y el compilador se encarga de garantizarlo. Basta pues con ejecutarlos 1 vez.

No obstante, conviene no engañarse con los bucles FOR y examinar su contenido. Si dentro del bucle se altera la variable de control, o el valor de alguna variable que se utilice en el cálculo del incremento o del límite de iteración, entonces eso es un bucle FOR con trampa.

También tiene "trampa" si contiene sentencias del tipo EXIT (que algunos lenguajes denominan BREAK) o del tipo RETURN. Todas ellas provocan terminaciones anticipadas del bucle.

Estos últimos párrafos hay que precisarlos para cada lenguaje de programación. Lo peor son aquellos lenguajes que permiten el uso de sentencias GOTO. Tampoco conviene confiarse de lo que prometen lenguajes como MODULA-2, que se supone que prohíben ciertas construcciones arriesgadas. Los compiladores reales suelen ser más tolerantes que lo que anuncia los libros.

Si el programa contiene bucles LOOP, o simplemente bucles con trampa, la única cobertura aplicable es la de ramas. El riesgo de error es muy alto; pero no se conocen técnicas sistemáticas de abordarlo, salvo reescribir el código.

Y en la práctica ¿qué hago?

Tanta definición acaba resultando un tanto académica e inútil.

En la práctica de cada día, se suele procura alcanzar una cobertura cercana al 100% de segmentos. Es muy recomendable (aunque cuesta más) conseguir una buena cobertura de ramas. En cambio, no suele hacer falta ir a por una cobertura de decisiones atomizadas.

¿Qué es una buena cobertura?

Pues depende de lo crítico que sea el programa. Hay que valorar el riesgo (o coste) que implica un fallo si éste se descubre durante la aplicación del programa. Para la mayor parte del software que se produce en Occidente, el riesgo es simplemente de imagen (si un juego fallece a mitad, queda muy feo; pero no se muere nadie). En estas circunstancias, coberturas del 60-80% son admisibles.

La cobertura requerida suele ir creciendo con el ámbito previsto de distribución. Si un programa se distribuye y falla en algo grave puede ser necesario redistribuirlo de nuevo y urgentemente. Si hay millones de clientes dispersos por varios países, el coste puede ser brutal. En estos casos hay que exprimir la fase de pruebas para que encuentre prácticamente todos los errores sin pasar nada por alto. Esto se traduce al final en buscar coberturas más altas.

Es aún más delicado cuando se trata de aplicaciones que involucran vidas humanas (aplicaciones sanitarias, centrales nucleares, etc.) Cuando un fallo se traduce en una muerte, la cobertura que se busca se acerca al 99% y además se presta atención a las decisiones atómicas.

También se suele perseguir coberturas muy elevadas (por encima del 90%) en las aplicaciones militares. Esto se debe a que normalmente van a ser utilizadas en condiciones muy adversas donde el tiempo es inestimable. Si un programa fallece, puede no haber una segunda oportunidad de arrancarlo de nuevo.

La ejecución de pruebas de caja blanca puede llevarse a cabo con un depurador (que permite le ejecución paso a paso), un listado del módulo y un rotulador para ir marcando por dónde se ha ido pasando. Esta tarea es muy trabajosa, pero puede ser automatizada. Hay compiladores que a la hora de generar código máquina dejan incrustado en el código suficiente código como para poder dejar un fichero (tras la ejecución) con el número de veces que se ha ejecutado cada sentencia, rama, bucle, etc.

Limitaciones

Lograr una buena cobertura con pruebas de caja blanca es un objetivo deseable; pero no suficiente a todos los efectos. Un programa puede estar perfecto en todos sus términos, y sin embargo no servir a la función que se pretende.

Por ejemplo, un Rolls-Royce es un coche que sin duda pasaría las pruebas más exigentes sobre los últimos detalles de su mecánica o su carrocería. Sin embargo, si el cliente desea un todo-terreno, difícilmente va a comprárselo.

Por ejemplo, se escribe una rutina para ordenar datos por orden ascendente, pero el cliente los necesita en orden decreciente; no hay prueba de caja blanca capaz de detectar la desviación.

Las pruebas de caja blanca convencen de que un programa hace bien lo que hace; pero no de que haga lo que se necesita.

3.2.2 Pruebas de caja negra

Sinónimos:

- pruebas de caja opaca
- pruebas Funciónales
- pruebas de entrada/salida
- pruebas inducidas por los datos

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas Funciónales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (en sentido general: teclado, pantalla, ficheros, canales de comunicaciones, etc.) Este comentario no obsta para que sean útiles en cualquier módulo del sistema.

Las pruebas de caja negra se apoyan en la especificación de requisitos del módulo. De hecho, se habla de "cobertura de especificación" para dar una medida del número de requisitos que se han probado. Es fácil obtener coberturas del 100% en módulos internos, aunque puede ser más laborioso en módulos con interfaz al exterior. En cualquier caso, es muy recomendable conseguir una alta cobertura en esta línea.

El problema con las pruebas de caja negra no suele estar en el número de funciones proporcionadas por el módulo (que siempre es un número muy limitado en diseños razonables); sino en los datos que se le pasan a estas funciones. El conjunto de datos posibles suele ser muy amplio (por ejemplo, un entero).

A la vista de los requisitos de un módulo, se sigue una técnica algebraica conocida como "clases de equivalencia". Esta técnica trata cada parámetro como un modelo algebraico donde unos datos son equivalentes a otros. Si se logra partir un rango excesivamente amplio de posibles valores reales a un

conjunto reducido de clases de equivalencia, entonces es suficiente probar un caso de cada clase, pues los demás datos de la misma clase son equivalentes.

El problema está pues en identificar clases de equivalencia, tarea para la que no existe una regla de aplicación universal; pero hay recetas para la mayor parte de los casos prácticos:

- si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
- si una entrada es booleana, hay 2 clases: si o no.
- los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Ejemplo: se utiliza un entero para identificar el día del mes. Los valores posibles están en el rango [1...31]. Así, hay 3 clases:

1. números menores que 1
2. números entre 1 y 31
3. números mayores que 31

Durante la lectura de los requisitos del sistema, se encontraran una serie de valores singulares, que marcan diferencias de comportamiento. Estos valores son claros candidatos a marcar clases de equivalencia: por abajo y por arriba.

Una vez identificadas las clases de equivalencia significativas en el módulo, se procede a coger un valor de cada clase, que no esté justamente al límite de la clase. Este valor aleatorio, hará las veces de cualquier valor normal que se le pueda pasar en la ejecución real.

La experiencia muestra que un buen número de errores aparecen en torno a los puntos de cambio de clase de equivalencia. Hay una serie de valores denominados "frontera" (o valores límite) que conviene

probar, además de los elegidos en el párrafo anterior. Usualmente se necesitan 2 valores por frontera, uno justo abajo y otro justo encima.

Limitaciones

Lograr una buena cobertura con pruebas de caja negra es un objetivo deseable; pero no suficiente a todos los efectos. Un programa puede pasar con holgura millones de pruebas y sin embargo tener defectos internos que surgen en el momento más inoportuno.

Por ejemplo, un PC que contenga el virus Viernes-13 puede estar pasando pruebas de caja negra durante años y años. Sólo falla si es viernes y es día 13; pero ¿a quién se le iba a ocurrir hacer esa prueba?

Las pruebas de caja negra convencen de que un programa hace lo que se quiere; pero no de que haga (además) otras cosas menos aceptables.

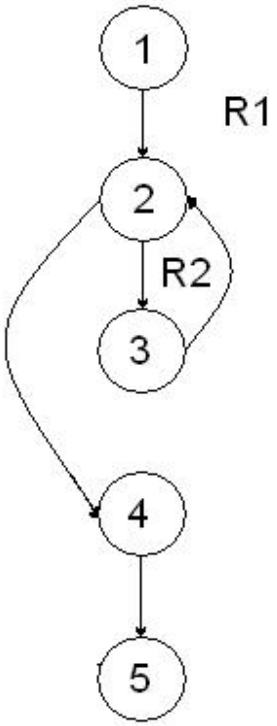
3.3 Aplicando caja blanca

Como se hace uso de la prueba de caja blanca con la utilización de la técnica del camino básico, se encaminará este apartado a aplicar los pasos necesarios para la aplicación de esta técnica. A continuación se presenta un segmento de código extraído de una función del algoritmo, la cual realiza una búsqueda de los rangos de temperaturas con las que cuenta el laboratorio para almacenar las muestras y carga un DropDownList con estos datos. Esta función no tiene parámetros y no devuelve ningún valor.

```
protected void CargarTemperatura()
1    {
1        TemperaturaNegocio tempNeg = new TemperaturaNegocio();
1        List<Temperatura> listTemp = tempNeg.ObtenerTodos();
1        List<RangoTemperatura> listTe = new List<RangoTemperatura>();
2        for (int i = 0; i < listTemp.Count; i++)
        {
3            RangoTemperatura temp = new RangoTemperatura();
3            temp.IdTemperatura = listTemp[i].IdTemperatura;
3              temp.TempMinMax = listTemp[i].TempMin.ToString() + "<>" +
listTemp[i].TempMax.ToString();
3            listTe.Add(temp);
        } 4
```

```
4         listT = listTe;
4         ddlTemperatura.DataSource = listT;
4         ddlTemperatura.DataBind();
    } 5
```

3.3.1 Grafo de flujo



3.3.2 Complejidad ciclomática

1era vía

V (G) = Nº de regiones del grafo

V (G) = 2

2do vía

V (G) = Nº de Aristas - Nº de Nodos + 2

V (G) = 5 – 5 + 2

V (G) = 2

3era vía

V (G) = Nº de Nodos Predicado + 1

V (G) = 1 + 1

V (G) = 2

El dato V (G) es la complejidad ciclomática del grafo de flujo, la misma define el número de caminos básicos por los cuales se recorre el algoritmo y de esta forma encontrar algún error en el funcionamiento de este.

3.3.3 Hallar los caminos independientes

Camino #1: 1, 2, 4, 5

Un objeto de negocio de temperatura

Una lista de temperaturas

Una lista de rangos de temperaturas

Resultado Esperado: Asumiendo que el listado de temperaturas que se extrajo de la base de datos estaba vacío entonces, a la hora de compilarse se llega al ciclo y se encuentra que listTemp.Count = 0 por lo tanto no entra y pasa al final del procedimiento, cargando el DropDownList vacio. Después en otra función se mostrara si esto sucede que no se puede almacenar muestras porque no hay condiciones en el laboratorio.

Camino #2: 1, 2, 3, 2, 4, 5

Un objeto de negocio de temperatura

Una lista de temperaturas

Una lista de rangos de temperaturas

Resultado Esperado: En este caso de prueba se asumió que la lista si tenia temperaturas. Por lo que se ejecuta mientras que i < listaTemp.Count, entonces se crea un rango de temperatura según el mínimo y máximo de temperatura para ese almacenamiento. Y después se carga el DropDownList con todos los rangos de temperatura, entre los cuales después el cliente escogerá en cual almacenar una muestra.

Se puede concluir que esta prueba ha demostrado que los caminos lógicos del software, estructura lineal y la implementación del software están correctamente. Teniendo en cuenta que el algoritmo escogido no era de mucha complejidad y la prueba de la caja blanca resultó satisfactoria porque los ya que los resultados fueron los esperados, entonces se puede afirmar que esta porción de código no posee ningún error. Se recomienda aplicar estas pruebas en una iteración durante su desarrollo como mínimo cinco veces y tratar de darle cobertura a casi el 100% del código para que así se considere el software con mayor calidad.

3.4 Aplicando caja negra

Proceso basado en el caso de uso Emitir Solicitud

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones
Se llenan los datos correspondientes. Paciente: (No se llena) Doctor: (No se llena) Prioridad: Emergencia Exámenes: (No se llena) Instrucciones Espec:(No) El servidor debe de estar conectado. Se da clic en el botón aceptar.	DatosPersona (Paciente) DatosPersona (Doctor) Exámenes	Se muestra dialogo de error. Debe seleccionar un Paciente	Satisfactorio	
Se llenan los datos correspondientes. Paciente: Alnair Reyes Pérez Doctor: (No se llena) Prioridad: Emergencia Exámenes: (No se llena) Instrucciones Espec:(No tiene) El servidor debe de estar conectado. Se da clic en el botón aceptar.	DatosPersona (Doctor) Exámenes	Se muestra dialogo de error. Debe Seleccionar un Doctor.	Satisfactorio	

Se llenan los datos correspondientes. Paciente: Alnair Reyes Pérez Doctor: Víctor Vázquez Estrada Prioridad: Emergencia Exámenes: (No se llena) Instrucciones Espec:(No tiene) El servidor debe de estar conectado. Se da clic en el botón aceptar.	Exámenes	Se muestra dialogo de error. Debe seleccionar al menos un examen.	Satisfactorio	
Se llenan los datos correspondientes. Paciente: Alnair Reyes Pérez Doctor: Víctor Vázquez Estrada Prioridad: Emergencia Exámenes: Hematología, Hemograma, Sangre total venosa. Instrucciones Espec:(No tiene) El servidor debe de estar conectado. Se da clic en el botón aceptar.		Se crea una nueva solicitud de análisis.	Satisfactorio	Se crea una nueva solicitud en la base de datos.

Se llenan los datos correspondientes. Paciente: Alnair Reyes Pérez Doctor: Víctor Vázquez Estrada Prioridad: Emergencia Exámenes: Hematología, Hemograma, Sangre total venosa. Instrucciones Espec: El paciente tiene hepatitis. El servidor debe de estar conectado. Se da clic en el botón aceptar.		Se crea una nueva solicitud de análisis.	Satisfactorio	Se crea una nueva solicitud en la base de datos.

Se llenan los datos correspondientes. Paciente: Alnair Reyes Pérez Doctor: Víctor Vázquez Estrada Prioridad: Urgencia Exámenes: Hematología, Hemograma, Sangre total venosa. Microbiología, Urocultivo, Orina. Instrucciones Espec: El paciente tiene hepatitis. El servidor debe de estar conectado. Se da clic en el botón aceptar.		Cambiar el brillo de la imagen moviendo Se crea una nueva solicitud de análisis.	Satisfactorio	Se crea una nueva solicitud en la base de datos.

Se puede concluir esta fase de prueba con la aplicación de la técnica de caja negra que el análisis al funcionamiento de los requisitos Funciónales del caso de uso Emitir Solicitud, están en perfecto estado. Y el diseño propuesto por los analistas y diseñadores del sistema se ajusta afinadamente a las necesidades presentadas cuando se va a emitir una solicitud de análisis.

Conclusiones

Como el Sistema de Información de Laboratorios es un módulo de un sistema mucho mayor (GeHos), en el futuro también se le realizaran pruebas de integridad y de aceptación, por ahora, usando las pruebas de unidades se comprobó que el sistema cuenta con las funcionalidades requeridas, siempre teniendo en cuenta que nunca se puede dar una cobertura del 100% a toda la funcionalidad, quedando siempre abierta la posibilidad de que existan algunos errores, los cuales se irán detectando en las siguientes iteraciones que se realicen en el desarrollo del proyecto.

Conclusiones

Con el estudio realizado, se ha confirmado que es verdaderamente necesaria la existencia de un sistema en los Laboratorios Clínicos de rutina que informatice el control de los procesos y dando una solución mas eficiente a los problemas actuales. Con este trabajo se ha logrado el objetivo trazado:

- Se hizo un estudio profundo de los procesos que se realizan en los laboratorios y la situación actual de estos.
- Se realizo un análisis detallado sobre los principales sistemas existentes en el mundo que tratan de darle una solución informática a dichos procesos.
- Se llevo a cabo un amplio estudio de la bibliografía relacionada con los temas de las herramientas, tendencias y tecnologías actuales.
- Se diseñó y elaboró un prototipo no funcional para presentar al usuario y determinar el cumplimiento de las funcionalidades estudiadas.
- Se desarrollo una versión beta del sistema, con calidad, portable, reutilizable con seguridad, escrito en un código legible y listo para proseguir con su desarrollo un los siguientes ciclos de desarrollo.
- Se le realizaron pruebas al sistema, permitiendo validar el desarrollo del mismo, quedando para una nueva etapa de desarrollo las pruebas de integración y de aceptación.

El sistema resultante logra un procesamiento óptimo de la información y disponibilidad de la misma en todo momento. Además será el punto de partida para la incorporación de nuevas funcionalidades en próximas iteraciones del desarrollo.

Recomendaciones

Se le recomienda a la dirección de la facultad 7 y la dirección del proyecto GeHos, continuar el trabajo y proseguir con los siguientes ciclos de desarrollos e iteraciones, para que en un final se logre una versión oficial del sistema, un producto acabado que se pueda instalar y explotar en los diferentes hospitales del país y otros países hermanos con la intención de incrementar el desarrollo alcanzado por el sistema nacional de salud.

Se le recomienda a la dirección de la facultad 7 y la dirección del proyecto GeHos, que se impartan cursos de superación para aquellos estudiantes y profesores que continúen el trabajo, así como que se incorpore de forma directa al trabajo un grupo de especialistas en el tema de los laboratorios clínicos con los que se lograría un mayor entendimiento del problema.

Referencias bibliográficas

1. Francisco Rojas Ochoa, Orígenes del movimiento de atención primaria de salud en Cuba [Disponible en: http://bvs.sld.cu/revistas/mgi/vol19_1_03/mgi10103.htm] (13/3/2007)
2. TIMSA software, Solución para laboratorios clínicos [Disponible en: http://www.timsa.com.mx/Páginas/07timlab.htm] (20/3/2007)
3. Elpidio Latorilla, CARE2X - El Proyecto, Entorno Integrado de Cuidados de la Salud [Disponible en: http://www.care2x.org/] (1/4/2007)
4. Healt Care International Services, CONLAB LIS® - Laboratorio [Disponible en: http://www.hcisonline.com/online/software.asp?id=6] (14/5/2007)
5. Microsoft, Detalles del :NET Framework [Disponible en: http://www.microsoft.com/spanish/msdn/comunidad/uni.net/PreGen/Uni_Framework/] (6/5/2007)
6. Jesse Liberty, O'Reilly, Programming C#, First Edition July 2001, ISBN: 0-596-00117-7, 680 pages (6/6/2007)
7. Ídem a la 6
8. Ídem a la 6
9. Ídem a la 6
10. Xavier Pey, ASP.NET, 11/06/2001 [Disponible en: http://www.netveloper.com/contenido2.aspx?IDC=39_0] (24/5/2007)
11. Juan Julián Merelo Guervos, Introducción a los servicios web y Microsoft .Net [Disponible en: http://geneura.ugr.es/~jmerelo/ws/] (28/5/2007)
12. Scott Short, Building XML Web Services for the Microsoft .NET Platform
13. JACOBSON, IVAR: RUMBAUGH, JAMES, BOOCH, GRADY: "El lenguaje unificado de modelado". Edit Addison-Wesley, 2000. (Cap. 11 p 281-302, cap. 16 p. 381-394).

Bibliografía

Discurso pronunciado por el Presidente de la República de Cuba, Fidel Castro Ruz, en la Tercera Graduación del Contingente del Instituto Superior de Ciencias Médicas de la Habana. Teatro "Carlos Marx". Ciudad de la Habana. 27 de agosto de 1990.

El Sistema Nacional de Salud de Cuba. Ramírez Márquez, Abelardo; Castell-Florit Serrate, Pastor; Mesa, Guillermo. ENSAP, 2003.

BRUEGGE, BERND, DUTOIT, ALLEN: Ingeniería de software: una perspectiva orientado a objetos, Edit. Pearson Educación, 2002.

JACOBSON, IVAR: RUMBAUGH, JAMES, BOOCH, GRADY: "El lenguaje unificado de modelado". Edit Addison-Wesley, 2000. (Cap. 11 p 281-302, cap. 16 p. 381-394).

LATORILLA, ELPIDIO: Care2x. Sistema Información Hospitalario, en http://care2x.ourproject.org, 3 de mayo del 2007.

MACDONALD, MATTHEW: Asp.Net Manual de Referencia. Edit. Mcgraw-Hill 2002.

MINSAP: Ministerios de salud publica, en http://www.dne.sld.cu/minsap/ Extraído el 30 de abril del 2007.

MRIDULA, PARIHAR: La Biblia De Asp.Net. Edit Anaya Multimedia-Anaya Interactiva, 2002.

PRESSMAN,ROGER: "Ingeniería del Software. Un enfoque práctico", Edit. McGraw-Hill/Interamericana de España, 2002.

REYNOSO, C. Y KICILLOF, N.: Estilos y Patrones en la Estrategia de Arquitectura de Microsoft, publicado en el año 2004, última actualización: 05/04/2007. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#24.

RECIO, FRANCISCO Y PROVENCIO, DAVID: Arquitectura básica de la plataforma .Net. Descripción del Framework y sus principales componentes: Lenguajes, biblioteca de clases y CLR. En http://www.desarrolloweb.com/articulos/1328.php. Extraído el 20 de marzo del 2007.

BIBLIOGRAFIA

WELICKI: Patrones de Fabricación: Fábricas de Objetos, en http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3624.asp, 2007.

ZELDMAN, JEFFREY: "Diseño con estándares Web". Edit. Anaya Multimedia-Anaya Interactiva, 2004.

Francisco Rojas Ochoa, Orígenes del movimiento de atención primaria de salud en Cuba [Disponible en: http://bvs.sld.cu/revistas/mgi/vol19_1_03/mgi10103.htm]

TIMSA software, Solución para laboratorios clínicos [Disponible en: http://www.timsa.com.mx/Paginas/07timlab.htm]

Elpidio Latorilla, CARE2X - El Proyecto, Entorno Integrado de Cuidados de la Salud [Disponible en: http://www.care2x.org/]

Healt Care International Services, CONLAB LIS® - Laboratorio [Disponible en: http://www.hcisonline.com/online/software.asp?id=6]

Microsoft, Detalles del :NET Framework [Disponible en: http://www.microsoft.com/spanish/msdn/comunidad/uni.net/PreGen/Uni_Framework/]

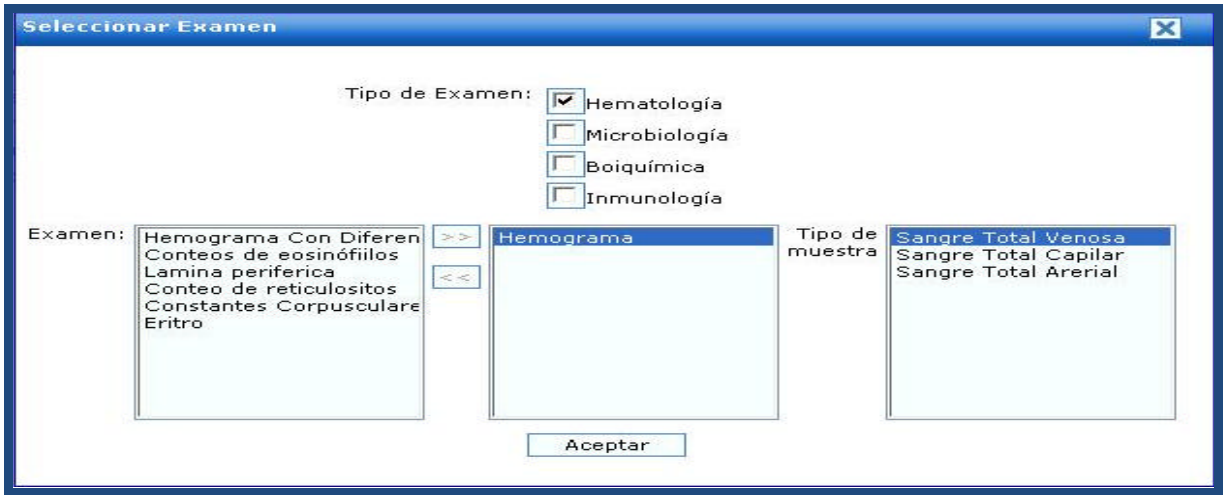
Jesse Liberty, O'Reilly, Programming C#, First Edition July 2001, ISBN: 0-596-00117-7, 680 páginas.

Juan Julián Merelo Guervos, Introducción a los servicios web y Microsoft .Net [Disponible en: http://geneura.ugr.es/~jmerelo/ws/] (28/5/2007)

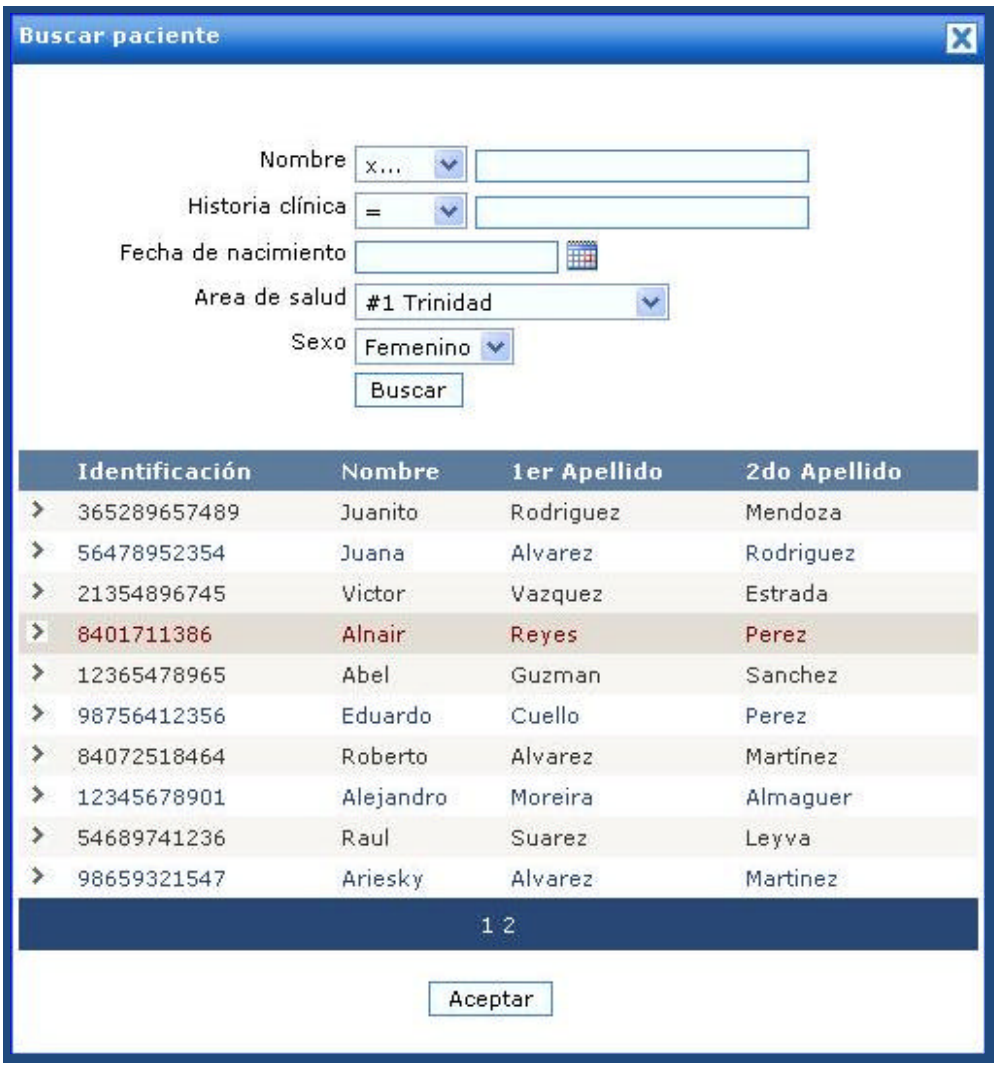
Anexo I: Prototipo no Funcional

Emitir solicitud

Seleccionar examen



Buscar Paciente



Bandeja de recepción

Recepcionar Análisis

Bandeja de toma de muestras

Toma de muestras

Almacenaje

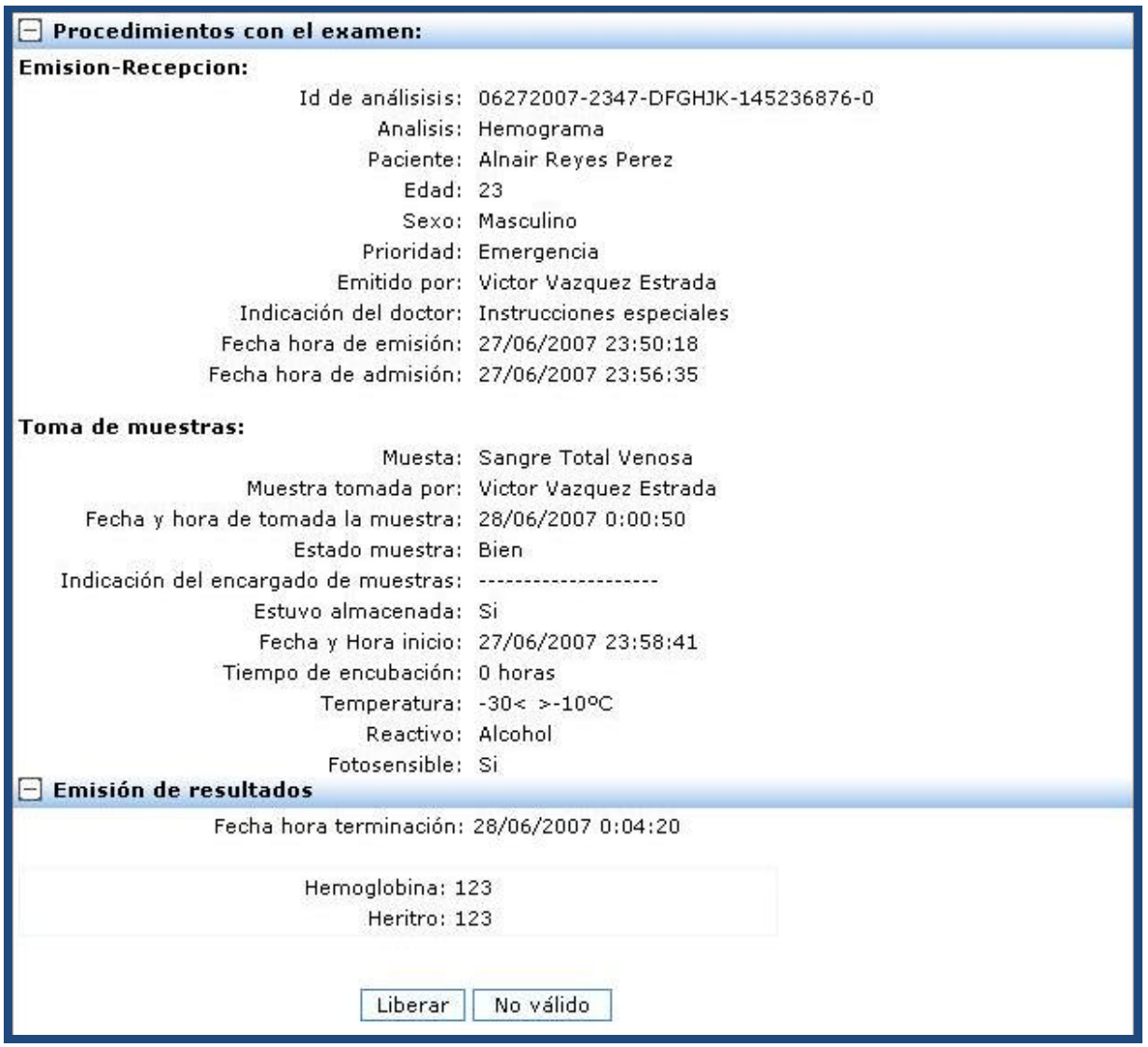
Bandeja de análisis

Emitir resultado

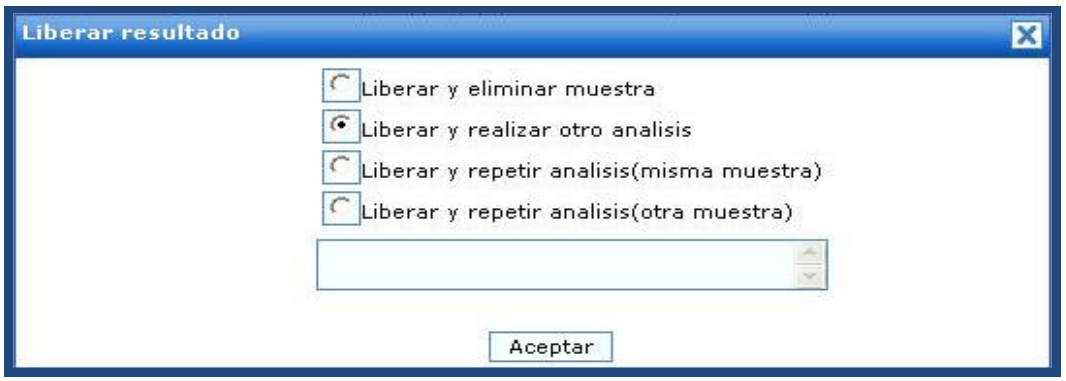
Bandeja de liberar

Análisis a liberar

Liberar análisis



Liberar



No valido