Universidad de las Ciencias Informáticas

Facultad #7

Título: Implementación del Módulo Bloque Quirúrgico del Sistema de Información Hospitalaria

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

Autores: Pastor López Gómez

Maikel García Hernández

Tutor: Ing. Ingrid Perez Tabrane

Consultante: Ing. Alfredo Morales Oliva

Ciudad de La Habana, Julio 2007

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los 4 días del mes de Julio del año 2007.

Maikel Garcia Hernandez

Firma del autor

Pastor López Gómez

Firma del autor

Ing. Ingrid Pérez Tabrane

Firma del tutor

PENSAMIENTO

"Los grandes espíritus siempre han encontrado la violenta oposición de las mentes mediocres. Estos últimos no pueden entender que un hombre no se someta irreflexivamente a los prejuicios hereditarios sino que emplee honestamente y con coraje su inteligencia."

Albert Einstein

DATOS DE CONTACTO

Ingrid Pérez Tabrane - ipereztabrane@uci.cu

Graduada de Ingeniería Informática en el año 2005 en el Instituto Superior Politécnico "Julio Antonio Echeverría" (CUJAE). Actualmente se desempeña como profesor universitario en la Universidad de las Ciencias Informáticas (UCI) donde ha impartido asignaturas como Teleinformática I, Teleinformática II y Seminario de Tesis. Posee categoría Instructor Recién Graduado. Se desempeñó por un periodo de 6 meses como líder del proyecto de Hospitales de la Facultad 7 de dicha universidad, el cual actualmente se convirtió en la temática productiva de Gestión Hospitalaria (GEHOS), en esta, se encuentra al frente del Módulo de Bloque Quirúrgico del Sistema de Información Hospitalaria desarrollado en la misma.

Alfredo Morales Oliva - amoraleso@uci.cu

AGRADECIMIENTOS

De Maikel:

A mis padres, por su confianza, esfuerzo y educación, que me han permitido llegar hasta aquí.

A mi hermano.

A mis abuelos, por su amor y por compartir mis sueños y alegrías.

A mi familia, por su apoyo en todos estos años.

A todos mis amigos y amigas.

A todos aquellos que contribuyeron con mi formación y en la realización de esta investigación.

A todos y cada uno.

Muchas Gracias.

De Pastor:

A mis padres, por su amor, sacrificio y entrega, que han contribuido en mi educación.

A mi familia, por su apoyo en todos estos años.

A mis compañeros de batalla de la FEU, por su amistad y por poder compartir mis discusiones y alegrías.

A todos mis amigos y amigas por apoyarme siempre.

A todos los profesores que contribuyeron en mi formación.

A mi compañero de proyecto Leonexy Oliva, por su ayuda en todo momento.

A mi novia, por su cariño y comprensión.

A todos y cada uno.

Muchas Gracias.

DEDICATORIA

A mi abuelo. Por su sabiduría.

A mis padres. Por su amor y comprensión.

A mi familia, en general.

De Maikel.

A mis padres. Por su amor y apoyo.

A mi familia, en general.

Al Comandante en Jefe Fidel Castro Ruz, por la idea tan genial, de crear esta universidad.

De Pastor.

RESUMEN

Con el presente trabajo se propone implementar un componente para la gestión de la información de los procesos vinculados con los servicios quirúrgicos que brindan los hospitales cubanos.

Para la implementación del componente se utilizó la plataforma .NET y el lenguaje de desarrollo del lado del servidor, ASP.NET. Como lenguaje de desarrollo del lado del cliente se usó JavaScript. También se utilizó la técnica de desarrollo AJAX, la herramienta de desarrollo Visual Studio 2005 y C# como lenguaje base de programación. La aplicación será compilada sobre framework .NET 2.0 para poder usar las bibliotecas de clases y el CLR de .NET. Algunos elementos de arquitectura presentes son: Arquitectura en tres capas, Arquitectura Cliente-Servidor, patrones de diseño, tales como, Fábrica Abstracta y Fabricación Pura y los patrones arquitectónicos: Arquitectura en Capas y Mapeo de Objetos.

Con el desarrollo de este componente se espera que pueda ser reutilizado por otros módulos, como, el Bloque Quirúrgico Oftalmológico, que brinde una interacción amigable entre el usuario y la interfaz y que los involucrados en los procesos que se llevan a cabo en los bloques quirúrgicos cuenten con una herramienta de trabajo capaz de gestionar de forma eficiente la información que se maneja en estos.

Palabras Claves: Componente, Bloque Quirúrgico Oftalmológico.

TABLA DE CONTENIDOS

LISTADO DE TABLAS

INTRODUCCIÓN

A lo largo de la historia, incluso desde tiempos en los que no existía la electricidad, el hombre siempre ha desarrollado instrumentos y herramientas que lo ayudan a mejorar y perfeccionar sus actividades productivas, influyendo en su modo de vida. A mediados del siglo XX el desarrollo científico y tecnológico hace posible que una nueva rama: la informática, comience un ascendente camino que llega hasta la actualidad donde se ha generalizado en todos los sectores de la sociedad.

En Cuba, se trabaja intensamente con el objetivo de utilizar las tecnologías de la información y la comunicación para apoyar la salud pública. Las acciones que se han emprendido en este sentido, parten de reconocer la importancia crucial de la revolución científico-técnica que se vive, pero sobre todo por priorizar el factor humano y adecuar estos avances a los problemas reales del país.

La sociedad cubana continuamente se nutre de los nuevos avances en las tecnologías de la información y las comunicaciones, la asistencia sanitaria se ha convertido paulatinamente en beneficiaria de las bondades que estas brindan. [1]

Premonizado por el Che desde el año 1962 y hecho realidad por el partido y gobierno cubano, el proceso de informatización de la sociedad cubana ha recibido una aceleración con la introducción de computadoras en múltiples esferas de la vida del país.

En el sector de la salud, la revolución lleva a cabo un fuerte programa para desarrollar la infraestructura tecnológica, a la par se facilita el aprendizaje y el intercambio de información científica profesional de alta calidad que permita dar total apoyo a las especialidades donde se desenvuelvan estos técnicos y profesionales. El país ha diseñado estrategias de bajo costo para hacer uso de la información y las comunicaciones como instrumentos y ponerlas a disposición del avance socialista. Se impone trabajar con mayor capacidad de investigación y desarrollo en estas áreas del conocimiento y adaptarlas a las condiciones de la sociedad donde existen potencialidades, fruto de su adecuada política educacional.

Como parte de los Programas de la Revolución, se ha creado la especialidad Estadísticas de Salud y la Universidad de Ciencias Informáticas (UCI), la cual fue creada con el objetivo de formar un personal con un elevado nivel técnico y profesional, que sirva como sostén al proceso de informatización que ocurre en la sociedad.

Actualmente, el país enfrenta la informatización de la sociedad apoyado en instituciones como la UCI. Esta le ha asignado a la facultad 7 algunas tareas encaminadas al logro de ese objetivo.

En ella, la producción se encuentra dividida en áreas temáticas, una de la cuales es Gestión Hospitalaria, cuya tarea es desarrollar sistemas que permitan la gestión de la información en los diferentes departamentos de un hospital, entre los que se encuentra el Bloque Quirúrgico con los servicios que lo conforman.

En Cuba, hasta el año 2005, según datos de los Registros administrativos de la Dirección Nacional de Planificación y Economía 2005, existían cerca de 248 hospitales. De ellos 35 son clínicos quirúrgicos. En estos, los procesos relacionados con las intervenciones quirúrgicas se tornan complicados, procesos tales como: atención al paciente, pruebas indicadas, planificación quirúrgica se llevan a cabo manualmente, donde se hace engorroso y se demora el trabajo de los médicos, en ocasiones también son usados algunos software que no cumplen con todos los requerimientos necesarios y que engloben los procesos que ahí se desarrollan. Teniendo en cuenta que los errores humanos están siempre presentes, y más cuando se trabaja con tanto llenado de planillas.

Dada la situación anterior el **problema** radica en ¿Cómo implementar un sistema informático que gestione la información de los procesos vinculados con las intervenciones quirúrgicas que tienen lugar en los hospitales del país?

El **objeto de estudio** se centra en el proceso de gestión de la información en los hospitales cubanos.

El **campo de acción** apunta al proceso de gestión de la información relacionada con las intervenciones quirúrgicas que tienen lugar en los hospitales cubanos.

Par dar solución al problema antes mencionado se propone como **objetivo general**: Implementar un sistema informático para la gestión de la información de los procesos vinculados con los servicios quirúrgicos que brindan los hospitales cubanos.

Para dar cumplimiento al objetivo general se han definido las siguientes **tareas de investigación:**

- Hacer un estudio de los sistemas similares existentes tanto nacionales como internacionales.
- Hacer un análisis crítico de la tecnología o lenguaje a utilizar.
- Analizar la Integración con otros componentes o partes del sistema.

- Realizar la implementación utilizando los patrones de diseño establecidos en la arquitectura del proyecto.
- Brindar una interfaz gráfica orientada al usuario y a las exigencias del mercado internacional.
- Garantizar la seguridad de los datos.
- Garantizar la interoperabilidad y flexibilidad del sistema.
- Realizar las pruebas necesarias para garantizar el correcto funcionamiento de la aplicación implementada.

Para dar cumplimiento a las tareas y objetivos planteados se usó en la investigación, dentro de los métodos empíricos: el análisis de documentos, el cual permitió determinar los problemas que presenta el sistema que usan hoy para la recogida de los datos quirúrgicos y además las mejoras que se le pueden hacer al mismo y se observo el posible futuro comportamiento de los planes del proyecto.

Los métodos teóricos tienen una especial importancia en el proceso de la investigación, utilizándose en la construcción y desarrollo de la teoría científica y en el enfoque general para abordar los problemas de la ciencia. Seguidamente se mencionan los métodos teóricos utilizados, el análisis histórico lógico, lo que permite estudiar de forma analítica la trayectoria histórica real y el desarrollo que han tenido todos los sistemas anteriores que se han utilizado en la parte quirúrgica de los Hospitales y el enfoque sistémico, donde se escoge el problema principal, se fracciona en sub-problemas y al final se procede a la integración de estos sub-problemas dándole una mayor calidad al proceso.

El contenido de este trabajo se estructura en tres capítulos:

El Capítulo I. Fundamentación teórica: Se realiza un estudio del estado del arte y se describen los principales aspectos de las herramientas más utilizadas en la producción de software a nivel mundial, además de las que se van a utilizar para la implementación de la aplicación, haciéndose comparaciones entre unas y otras en cuanto a ventajas y desventajas.

El Capítulo II. Descripción y análisis de la solución propuesta. Se realiza una valoración crítica del diseño propuesto por los analistas, un análisis sobre componentes, módulos o implementaciones que podrían ser reutilizadas para dar solución al problema propuesto, así como la descripción de las clases implementadas para dar solución al problema.

El Capítulo III. Validación de la solución propuesta. En este capítulo se realizara un análisis y descripción de los posibles “Test de unidades” para validar la solución propuesta. Se realiza una descripción de los valores utilizados para los test, así como una evaluación de la ejecución y los resultados obtenidos.

Cada capítulo es iniciado por una breve introducción donde se dan a conocer los temas que se desarrollarán durante el mismo. Finaliza con las conclusiones, en las que se plantean los resultados obtenidos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se realiza un análisis detallado del estado del arte de las distintas técnicas de programación que existen a nivel internacional, nacional y de la Universidad. Se hace un estudio de los lenguajes de programación, tecnologías y metodologías relacionadas con dichas técnicas, así como las plataformas de desarrollo que la soportan. Se realiza un estudio crítico y valorativo de la técnica de programación, plataforma y librerías usadas para el desarrollo del sistema, teniendo en cuenta las necesidades y las características del entorno donde se aplicará la solución propuesta.

1.1 Comportamiento de los procesos de gestión de la información hospitalaria

En al segunda mitad del siglo XX, se perfilan los primeros sistemas de información médica que, posteriormente, habrán de dar lugar a los sistemas de información hospitalaria, tan indispensables en la actualidad. Su impacto en las instituciones de salud es notable, ya que buscan elevar la calidad de la atención del paciente, de los servicios brindados y aplicar la información obtenida a las áreas de la investigación, la clínica, la docencia, la administración y desde luego abatir costos y elevar la productividad. [2]

De los sistemas de información hospitalaria (SIH) existen disímiles productos y versiones pertenecientes a varias empresas de desarrollo de software. En Cuba, no existe la suficiente difusión e intercambio de experiencias como para realizar y estandarizar un producto que cubra las expectativas de los centros de salud existentes. El “Galen Hospital” y el “Medisys” son ejemplos de algunos sistemas implantados y funcionando, pero con algunas limitaciones de integración y estandarización.

Estos sistemas, en su mayoría diseñados e implementados usando el estilo arquitectónico cliente/servidor en tres capas, han sido desarrollados utilizando software privativo. Lo que eleva en gran medida los costos de producción e implantación, además de imposibilitar la reutilización de estas experiencias en nuevas versiones, atentando entre otras cosas contra la estandarización. Se puede decidir dentro de estas posibilidades, por aplicaciones web y de escritorio, aunque dentro de estas se presentan algunos impedimentos.

Los sistemas que poseen interfaces web, presentan en algunos casos incompatibilidades con los navegadores que en la actualidad encabezan la lista de los más utilizados, o aquellos que posibilitarían la explotación de la aplicación en clientes que usen sistemas operativos basados en la plataforma GNU-

LINUX. Sumándose aquellos sistemas consistentes en aplicaciones de escritorio compatibles solo sistemas operativos Windows.

1.2 Sistemas existentes vinculados al campo de acción

La meta es construir un Bloque Quirúrgico (BQ) estandarizado donde el paciente sea el más beneficiado, y los profesionales de la salud encuentren en este sistema un recurso idóneo, amigable y flexible que responda a las necesidades de información de la institución hospitalaria o de salud.

Cuba cuenta con el Sistema de Información Hospitalaria (SIH), pero este no presenta todos los requerimientos necesarios. Solo tiene implementado una Solución Quirúrgica en Visual Basic 6. A nivel internacional se encuentra el *Nuring Information System* (NIS) implementado en Delphi 7. Sistema de gran actividad e interacción con los diferentes sistemas no solo el Clínico, sino también con el Administrativo y con el de Laboratorios. Integra al personal de enfermería a la sistematización de procesos, agilizando y liberando la carga de trabajo para la atención con calidad al paciente. Cuenta con los módulos Administración Quirófano y Atención al Paciente. [3]

Un ejemplo fehaciente de un sistema que ha dado los primeros pasos en el sentido propuesto y que contiene un componente para la parte Quirúrgica de los Hospitales llamado “Sala de Operaciones” , es el Care2X, sistema desarrollado desde el 2002 por un grupo mundial de programadores utilizando herramientas libres y en el cual se evidencia un adelanto a la solución que se persigue, este sistema utiliza el protocolo de intercambio de datos *Health Exchange Protocol* (HXP), este protocolo está siendo usado por Care2x para comunicarse con otras aplicaciones independientemente de sus plataformas. HXP hace intercambio de datos entre aplicaciones para la salud, además de ser simple de implementar y de comprender, independiente de plataforma, libre o de muy bajo costo, confiable, seguro con autenticación, encriptado con protocolo SSL, flexible, transparente, libre de restricciones geográficas y código libre. Hasta el momento, está compuesto por cuatro componentes principales. Cada uno de estos componentes también puede funcionar de manera individual. [4]

1.3 Tendencias, Herramientas y Tecnologías

Estas ayudan al equipo de desarrollo a diseñar y construir la aplicación. Son de vital importancia para el buen desempeño de la aplicación, la selección de las mismas está determinada por las tecnologías en que se desarrolle la aplicación.

1.3.1 Técnicas de programación

1.3.1.1 Programación no Estructurada

Esta programación consiste en secuencias de instrucciones donde los saltos y el fin de programa no siguen ninguna estructura. Los saltos apuntan a cualquier punto del código lo que ocasiona que el algoritmo termine siendo un ovillo indescifrable. **(Ver Anexo no. 6)**

1.3.1.2 Programación Procedimental

Es un tipo de programación estructurada en donde el código se divide en porciones llamadas "procedimientos" o "funciones". **(Ver Anexo no. 7 y 8)**

1.3.1.3 Programación Modular

Está basada en la técnica de diseño descendente, que consiste en dividir el problema original en diversos sub-problemas que se pueden resolver por separado, para después recomponer los resultados y obtener la solución al problema. **(Ver Anexo no. 9)**

1.3.1.4 Programación Orientada a Objetos (POO)

Es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan *estado* (es decir, datos), comportamiento (esto es, procedimientos o *métodos*) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar. De esta forma, un objeto contiene toda la información, (los denominados atributos) que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases (e incluso entre objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos). A su vez, dispone de mecanismos de interacción (los llamados métodos) que favorecen la comunicación entre objetos (de una misma clase o de distintas), y en consecuencia, el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan (ni deben separarse) información (datos) y procesamiento (métodos). [5] **(Ver Anexo no. 10)**

1.3.1.5 Programación orientada a componentes (POC)

La POC nace con el objetivo de construir un mercado global de componentes de softwares, cuyos usuarios son los propios desarrolladores de aplicaciones que necesitan reutilizar componentes ya hechos y probados para construir sus aplicaciones de forma más rápida y robusta. Las entidades básicas de la POC son los componentes, es decir cajas negras que encapsulan cierta funcionalidad y que son diseñadas para formar parte de ese mercado global de componentes, sin saber ni quién los utilizará, ni cómo, ni cuándo. Los usuarios conocen acerca de los servicios que ofrecen los componentes a través de sus interfaces y requisitos, pero normalmente ni quieren ni pueden modificar su implementación.

Aparece como una variante natural de la POO para los sistemas abiertos, en donde la POO presenta algunas limitaciones; por ejemplo, la POO no permite expresar claramente la distinción entre los aspectos computacionales y meramente composicionales de la aplicación, no define una unidad concreta de composición independiente de las aplicaciones (los objetos no lo son, claramente), y define interfaces de demasiado bajo nivel como para que sirvan de contratos entre las distintas partes que deseen reutilizar objetos.[6]

1.3.1.6 La programación orientada a servicios

La programación orientada a servicios es un complemento de la programación orientada a objetos e implementa mejoras a esta última, producto de la experiencia acumulada en la última década, favorece la instalación de una solución e interoperabilidad entre sistemas heterogéneos.

La industria del desarrollo de software se encuentra actualmente en un estadío de transición hacia un nuevo paradigma de programación: la llamada Arquitectura Orientada a Servicios (SOA), por sus siglas en inglés). Es importante recalcar que SOA no reemplaza a POO, sino que la complementa introduciendo mejoras. [7]

SOA Vs POO

Una de las diferencias fundamentales entre SOA y POO es la manera en la que ambas definen una “aplicación”. POO determina que una aplicación está compuesta de clases interdependientes, mientras que SOA considera que una aplicación está compuesta por un conjunto de servicios autónomos.

SOA, a diferencia de las arquitecturas de objetos distribuidos como J2EE, refleja más fielmente los procesos y relaciones del mundo real; es decir, SOA representa una manera más simple y natural de modelar y construir software que soluciona problemáticas de negocios del mundo real.

Los sistemas orientados a servicios, en cambio, son diseñados con un bajo nivel de acoplamiento que facilita la implementación de cambios y estos servicios pueden ser desarrollados en cualquier lenguaje corriendo en diferentes plataformas.

Desde el punto de vista de un usuario, la diferencia entre utilizar servicios y utilizar componentes tradicionales es imperceptible. Desde el punto de vista arquitectónico, en cambio, se puede decir que los servicios a diferencia de los componentes son autónomos, encapsulan sus propios datos y no forman parte de la aplicación, sino que son utilizados por ella. Estos servicios pueden ser generados por distintos equipos de desarrollo, en diferentes plataformas y en diferentes espacios y tiempos; es decir, que una aplicación puede ir creciendo y aumentando su funcionalidad a medida que se construyan nuevos servicios que no necesariamente deben estar bajo el control del equipo de desarrollo de la aplicación, por lo que es esencial que entre las aplicaciones y los servicios que ellas consumen, exista un mínimo acoplamiento.

Se pueden utilizar servicios disponibles en Internet para, por ejemplo, permitirle a los usuarios utilizar algún motor de búsqueda como *Google*; pero también puede que la aplicación sea un conjunto de servicios, por ejemplo, el servicio de crear un cliente, etc., de esa manera no solo se está adhiriendo al concepto de SOA sino, que se está dejando abierta una puerta para que futuras aplicaciones hagan uso de esa funcionalidad con un mínimo de esfuerzo.

Sobre la capa de componentes de negocio tradicionales se encuentra una capa llamada Interfaces de Servicios. Más allá de la nomenclatura, lo importante de esta capa es que utiliza los conceptos expuestos previamente, transformando los componentes desarrollados con POO en servicios autónomos. También hay una capa de Servicios al mismo nivel que la capa de Orígenes de datos; esta capa indica que los servicios pueden hacer uso de servicios preexistentes tratándolos como un origen de datos más.

Los servicios son utilizados por una aplicación cliente que les envía mensajes respetando un determinado contrato, pero internamente esos servicios utilizan los conceptos de POO.

Implementando una Arquitectura orientada a servicios tiene como ventajas:

Es altamente interoperable entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen. Permite que los componentes de software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar; así, un servicio C# podría ser usado por una aplicación Java.

Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento. Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.

Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados. Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar.

1.3.2 Lenguajes de programación

Los lenguajes de programación son creados por el hombre para poder comunicarse con las computadoras. De esta forma un lenguaje de programación consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.

La programación Web, parte de las siglas WWW, que significan *World Wide Web* o telaraña mundial. Es un sistema de navegador web para extraer elementos de información llamados "documentos" o "páginas web". Puede referirse a "una web" como una página, sitio o conjunto de sitios que proveen información por los medios descritos, o a "la Web", que es la enorme e interconectada red disponible prácticamente en todos los sitios de Internet. Ésta es parte de Internet, siendo la *World Wide Web* uno de los muchos servicios ofertados en la red Internet.

Una de las cualidades de Internet de los restantes medio de comunicación es la personalización de la información del usuario mediante los diversos lenguajes de programación, estos lenguajes se clasifican en dos grupos importante que son. Lenguajes del lado del cliente y lenguajes del lado del servidor.

1.3.2.1 Lenguajes del lado del cliente

HTML

El HTML, acrónimo inglés de *HyperText Markup Language* (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo *Internet Explorer*, Opera,

Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos, es una aplicación de SGML conforme al estándar internacional ISO 8879. [8]

JavaScript

Es un lenguaje interpretado, es decir, que no requiere compilación, orientado a las páginas web. Dirigido por eventos, estará listo para actuar en cuanto un evento (un click en un botón, por ejemplo) sea ejecutado, implementa una sencilla interfaz de objetos/propiedades/métodos. Se integra dentro del código HTML de las páginas Web, se ejecuta en el navegador al mismo tiempo que las sentencias van descargándose junto con el código HTML. Brinda rapidez a la aplicación web ha la hora de las validaciones de los formularios. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. [9]

Visual Basic Script

VBScript (abreviatura de *Visual Basic Script Edition*) es un lenguaje interpretado. Su sintaxis refleja su origen como variación del lenguaje de programación *Visual Basic*. Ha logrado un apoyo significativo por parte de los administradores de Windows como herramienta de automatización, ya que, conjunta y paralelamente a las mejoras introducidas en los sistemas operativos windows donde opera fundamentalmente, permite más margen de actuación y flexibilidad que el lenguaje *batch* (o de proceso por lotes) desarrollado a finales de los años 1970 para el MS-DOS. [10]

XSLT

XSL Transformaciones (XSLT) que parte de *The Extensible Stylesheet Language Family* (XSL), es un estándar que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML. Las hojas de estilo (aunque el termino de hojas de estilo no se aplica sobre la función directa del XSLT) XSLT realizan la transformación del documento utilizando una o varias reglas de plantilla unidas al documento fuente a transformar, esas reglas de plantilla alimentan a un procesador de XSLT, el cual realiza las transformaciones deseadas colocando el resultado en un archivo de salida o, como en el caso de una página web, directamente en un dispositivo de presentación, como el monitor de un usuario.[11]

1.3.2.2 Lenguajes del lado del servidor

ASP.NET

ASP.NET es un conjunto de tecnologías de desarrollo de aplicaciones web comercializado por Microsoft. Es usado por programadores para construir sitios web domésticos, aplicaciones web y servicios XML. Forma parte de la plataforma .NET de Microsoft y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET es la plataforma unificada de desarrollo Web que proporciona a los desarrolladores los servicios necesarios para crear aplicaciones Web empresariales. [12]

- Mejor rendimiento, eficacia y flexibilidad**.** ASP.NET es un código de Common Language Runtime compilado que se ejecuta en el servidor. A diferencia de sus predecesores, ASP.NET 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. 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. Es también independiente del lenguaje, por lo que puede elegir el lenguaje que mejor se adapte a la aplicación o dividir la aplicación en varios lenguajes.
- 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. La edición WYSIWYG, los controles de servidor de arrastrar y colocar y la implementación automática son sólo algunas de las características que proporciona esta eficaz herramienta.
- Facilidad de uso**.** ASP.NET emplea un sistema de configuración jerárquico, basado en texto, que simplifica la aplicación de la configuración al entorno de servidor y las aplicaciones Web. Debido a que la información de configuración se almacena como texto sin formato, se puede aplicar la nueva configuración sin la ayuda de herramientas de administración local. No se requiere el reinicio del servidor, ni siquiera para implementar o reemplazar el código compilado en ejecución.
- Escalabilidad y disponibilidad**.** ASP.NET se ha diseñado teniendo en cuenta la escalabilidad, con características diseñadas específicamente a medida, con el fin de mejorar el rendimiento en entornos agrupados y de múltiples procesadores. Además, 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.

- 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

PHP

PHP es un lenguaje de programación usado frecuentemente para la creación de contenido para sitios web con los cuales se puede programar las páginas html y los códigos de fuente. PHP es un acrónimo recursivo que significa "*PHP Hypertext Pre-processor*" (inicialmente PHP Tools, o, *Personal Home Page Tools*), y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios web. [13]

Java

Java es un lenguaje de programación orientado a objetos desarrollado por *Sun Microsystems* a principios de los años 1990. Las aplicaciones Java están típicamente compiladas en un bytecode,. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

Esta distinción entre los lenguajes ha sido necesaria debido a que el protocolo http es un protocolo sin estado (*state less*), no guarda información sobre conexiones anteriores y al finalizar la transacción los datos se pierden, cada petición/respuesta es una operación distinta, por lo que la Web trabaja en modo desconectado; o sea, un usuario a través de un navegador hace una petición de una página Web a un Servidor Web (*Request*), el Servidor obtiene la petición, la procesa y le envía la respuesta al Cliente (*Response*), este hace la recepción y se desconecta. Por esto se popularizaron las *cookies*, que son pequeños ficheros guardados en el propio ordenador que puede leer un sitio web al establecer conexión con él, y de esta forma reconocer a un visitante que ya estuvo en ese sitio anteriormente. Gracias a esta identificación, el sitio web puede almacenar gran número de información sobre cada visitante, ofreciéndole así un mejor servicio. [14]

C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. Su

sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes (más notablemente de Delphi y Java). C# fue diseñado para combinar el control a bajo nivel de lenguajes como C y la velocidad de programación de lenguajes como Visual Basic. [15]

C#, como parte de la plataforma .NET, está normalizado por ECMA desde diciembre de 2001 (ECMA-334 "Especificación del Lenguaje C#"). El 7 de noviembre de 2005 acabó la beta y salió la versión 2.0 del lenguaje que incluye mejoras tales como tipos genéricos, métodos anónimos, iteradores, tipos parciales y tipos anulables. Ya existe la versión 3.0 de C# en fase de beta destacando los tipos implícitos y el LINQ (Language Integrated Query). [16]

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

1.3.3 XML y Javascript Asíncrono (Ajax)

AJAX, acrónimo de *Asynchronous JavaScript And XML* (JavaScript y XML asíncronos, donde XML es un acrónimo de eXtensible Markup Language), 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. [17]

1.3.4 Metodologías de desarrollo de software

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, como resultado se obtendrá clientes y desarrolladores insatisfechos.

El Proceso Racional Unificado o **RUP** (Rational Unified Process) es un proceso de desarrollo de software (conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software) que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de amplitud y diferentes tamaños de proyectos. [18]

El Proceso Unificado está basado en componentes. Utiliza el lenguaje unificado de modelado (UML) para preparar todos los esquemas de un sistema de software. De hecho, UML es una parte esencial de RUP, sus desarrollos fueron paralelos. No obstante los verdaderos aspectos definitorios del proceso unificado se resumen en tres fases claves: dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

El Proceso Racional Unificado, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas.

MSF

Microsoft Solutions Framework (MSF) es una flexible e interrelacionada serie de conceptos, modelos y prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. Originalmente creado en 1994 para conseguir resolver los problemas a los que se enfrentaban las empresas en sus respectivos proyectos, se ha convertido posteriormente en un modelo práctico que facilita el éxito de los proyectos tecnológicos. [19]

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación

Xp

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizadas para proyectos de corto plazo y corto equipo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto [20]. Esta metodología se basa en:

- **Pruebas Unitarias:** se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantando en algo hacia el futuro, se pueden hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.
- **Refabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

- **Programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

¿Qué es lo que propone XP?

Empieza en pequeño y añade funcionalidad con retroalimentación continua.

El manejo del cambio se convierte en parte sustantiva del proceso.

El costo del cambio no depende de la fase o etapa.

No introduce funcionalidades antes que sean necesarias.

1.4 Plataformas de desarrollo

Dentro de las más utilizadas se encuentran:

.NET: La plataforma .NET no es más que un conjunto de tecnologías para desarrollar y utilizar componentes que permitan crear formularios web, servicios web y aplicaciones Windows. [21]

J2EE: tecnología Java 2 Enterprise Edition (J2EE) proporciona una completa y potente plataforma orientada al desarrollo de aplicaciones corporativas distribuidas y a los servicios web. J2EE integra un conjunto de APIs (Application Programming Interface), frameworks y patrones de programación que permiten responder de una forma robusta y flexible a todas estas demandas. Para cada una de las capas de la aplicación, capa de presentación, capa de negocio y capa corporativa se aportan muy buenas soluciones. [22]

La tecnología J2EE también aporta un enfoque estándar para el desarrollo de componentes web para aplicaciones menos complejas, que podrán ser rehusados cuando se quiera escalar la aplicación.

MONO: Es la plataforma de desarrollo de software libre basada en .NET que permite a los desarrolladores de software construir aplicaciones GNU/Linux y multiplataforma con una productividad sin precedentes. La implementación de la plataforma .NET de Mono se basa en los estándares de ECMA C# y la infraestructura del lenguaje común. Mono incluye las herramientas de desarrollo y la infraestructura necesarias para crear aplicaciones del cliente y el servidor. [23]

1.5 Tecnologías y metodologías utilizadas

Constituye un objetivo fundamental de los diseñadores de software alcanzar y mantener un nivel técnico acorde con el desarrollo actual en la automatización de la información para la gestión de cualquier proceso a desarrollar, para lo cual es necesario hacer un estudio detallado da las tecnologías a utilizar y las posibilidades de desarrollo que estas brindan, así como los conceptos ligados a estas. A continuación en esta sesión se describe los principales conceptos, tecnologías y herramientas propuestas para el desarrollo de la solución tratada en el trabajo.

Para dar solución al problema y luego de haber analizado un grupo de tecnologías y lenguajes candidatos, se decidió por los arquitectos del sistema el uso de la plataforma .NET y por ende el lenguaje de desarrollo del lado del servidor ASP.NET por su gran rendimiento, compatibilidad con herramientas de primer nivel, eficacia, flexibilidad, simplicidad, facilidad de uso, escalabilidad, disponibilidad y seguridad. Como lenguaje de desarrollo del lado del cliente se usa JavaScript.

Se emplea la metodología AJAX y para el desarrollo la herramienta Visual Studio 2005. También se utiliza C# como lenguaje de programación, ya que es el lenguaje sobre la plataforma Mono donde se puede crear aplicaciones multiplataforma de código abierto que funciona perfectamente sobre los sistemas Linux, Solaris, Mac OS X, Windows, y Unix. Se utiliza gestor de base de datos PostgresSQL.

La aplicación será compilada sobre framework .NET[1] para poder usar las bibliotecas de clases y el CLR de .NET, pero en un futuro cuando se implemente una siguiente versión para software libre deberá ser compilada en sobre framework Mono y como herramienta de implementación se usará el SharpDevelop y como gestor de base de datos se seguirá usando PostgresSQL debido a que este último pertenece a la familia de software libre.

1.6 Plataforma y librerías utilizadas para el desarrollo del módulo

Para el desarrollo del módulo se utiliza la plataforma .NET ya que está diseñada para que se puedan desarrollar componentes de software utilizando cualquier lenguaje de programación, de forma que el

[1] Consultar Anexo no. 11

código fuente de un lenguaje pueda utilizarse desde cualquier otro de la manera más transparente posible (utilizando servicios web como middleware). Además tiene grandes ventajas como son:

- Interoperabilidad multilenguaje, que permite que el código pueda ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio (MSIL), lo que permite combinar, de ser necesario, códigos de diferentes lenguajes.
- Compilación just-in-time, que permite que el compilador JIT incluido en el Framework compile el código intermedio (MSIL), generando el código máquina propio de la plataforma. Esto ofrece mayor rendimiento de la aplicación ganando en rapidez. [24]
- Código administrado: El CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente. [25]

Por otro lado las librerías utilizadas son las relacionadas con las bibliotecas de clases de .Net las cuales se encuentran incluidas dentro del Framework el cual es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Un ejemplo específico de estas librerías es system.web, system.windows.forms, system.data, system, etc.

1.7 Elementos de arquitectura presentes

¿Que es una arquitectura?

Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización.

Los principales componentes que distinguen la arquitectura son:

- Presentación:

  Esta capa contiene las interfaces necesarias para que el usuario y el sistema intercambien toda la información necesaria para el proceso de gestión, compuesta por páginas WEB ASP.NET. Interactúa con la capa inferior, mediante invocación de los métodos que conformar la lógica del negocio, produciendo un intercambio de objetos.

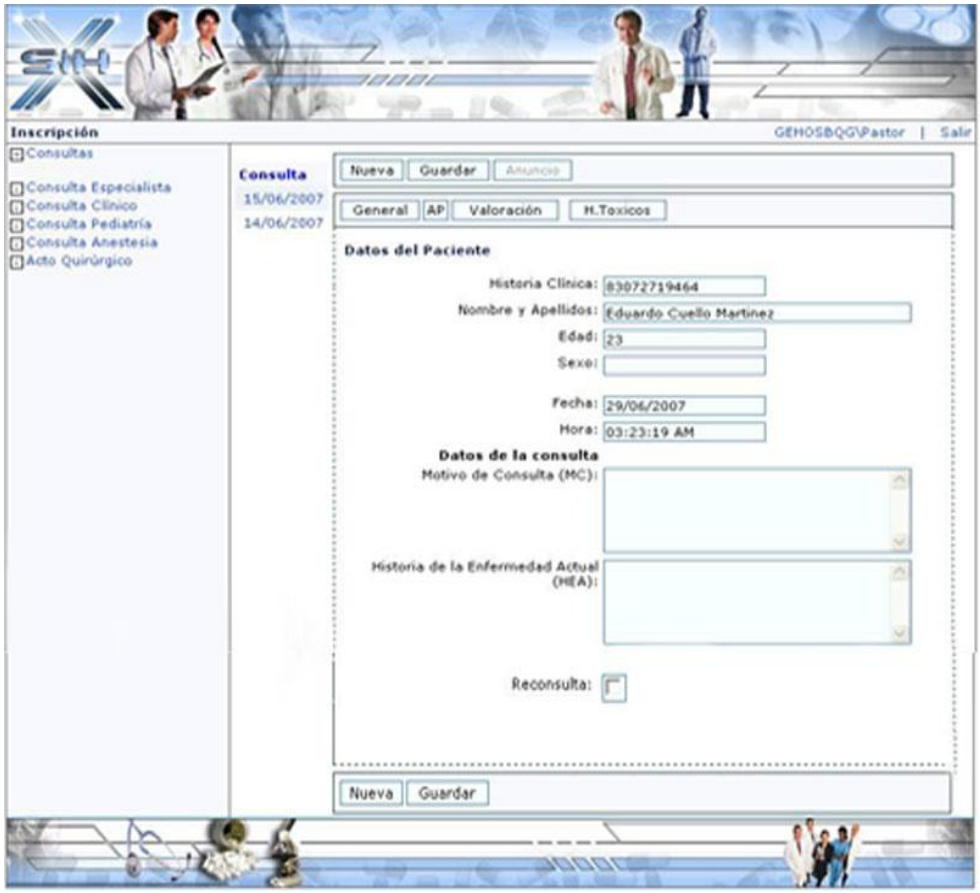


Fig. No.1 Interfaz de Usuario

- Negocio:

  Esta capa se divide en dos subcapas encargadas de resolver la lógica del negocio.

  Clases del Negocio: Almacena todas las clases encargadas de representar la lógica del negocio, y se controla la seguridad en cada invocación de los métodos. Interactúa con la subcapa Middleware invocando los métodos definidos en las clases Repositorios, el intercambio consiste al igual que en el caso anterior en colecciones de objetos.

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

  Repositorios: Los repositorios son los encargados de manejar las colecciones de Objetos Comunes (Entidades de la BD representados como objetos) y realizar operaciones sobre ellas.

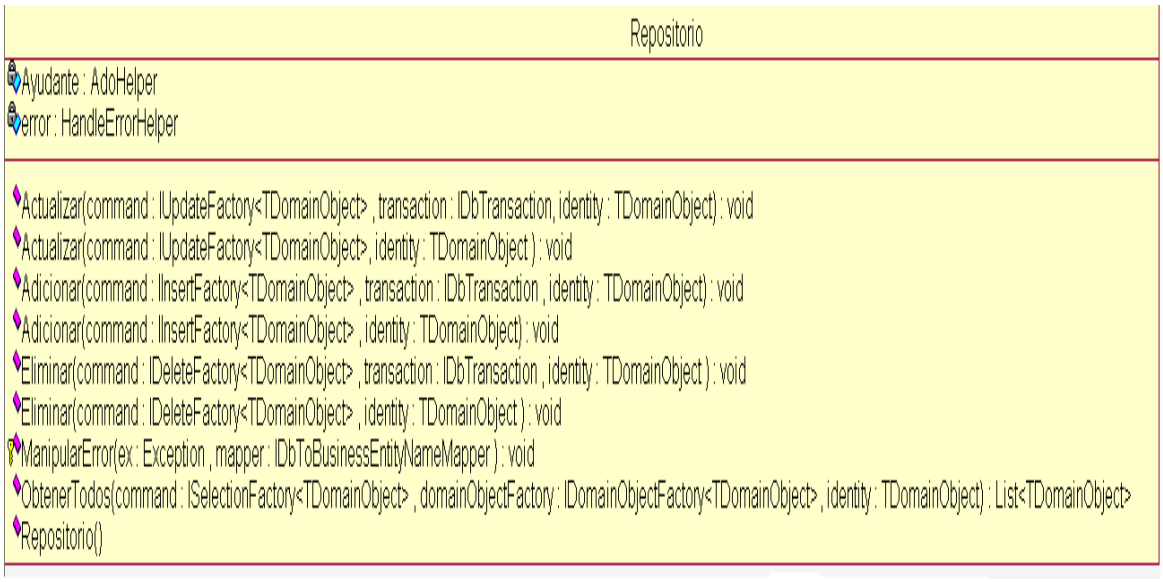


Fig. No. 2 Clase Repositorio

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

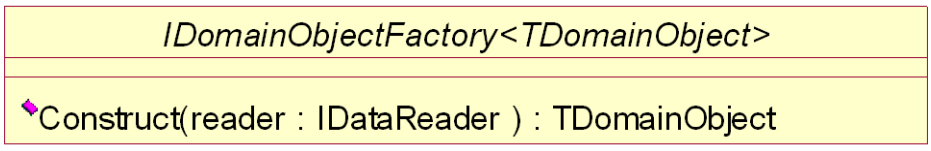


Fig. No. 3 Clase Interfaz IDomainObjectFactory

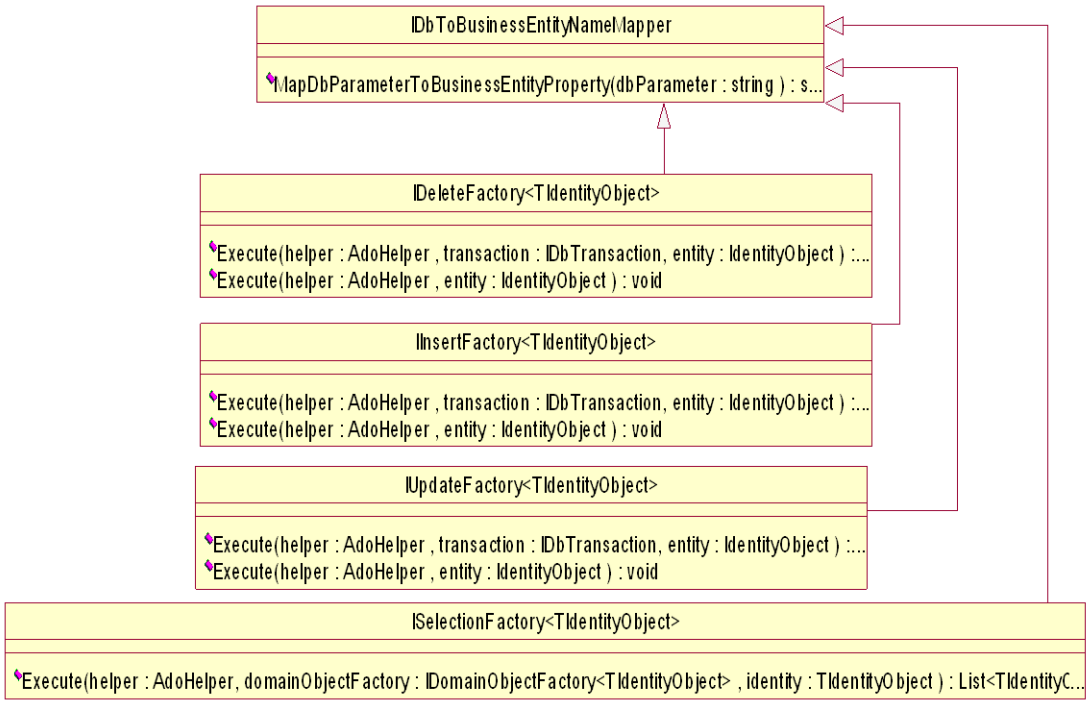


Fig. No. 4 Relación entre las diferentes Fábricas y la clase DbToBusinessEntityNameMapper

- Acceso a datos: Ubicada en el servidor de Bases de Datos, está compuesta por el conjunto de funciones programadas en lenguaje NpgSql que se enlazan con el middleware para ejecutar las solicitudes sobre el servidor de Bases de Datos. Engloban toda la lógica de acceso a datos.

- ESB SOA GeHos: Conjunto de Servicios WEB que cubren las funcionalidades exigidas por la lógica de negocio; utilizados para facilitar la integración con sistemas externos ó módulos del sistema, que no se puedan conectar directamente con el negocio de los restantes módulos por problemas de configuración de la red o localizaciones de estos. La información facilitada por estos consisten en mensajes HL7 sobre XML construidos y revisados utilizando Chameleon.

Los conectores, son las formas de comunicación entre los componentes o elementos definidos, es la forma en que está representada la información que fluye entre estos.

- Presentación – Negocio.

Las funcionalidades que brinda el negocio son accedidas mediante invocación de métodos y las respuestas de estos constituyen colecciones de objetos.

- Negocio – Middleware

  Las funcionalidades del Middleware son accedidas mediante invocación de métodos pertenecientes a las clases Repositorios y las respuestas de estos constituyen colecciones de objetos.

- Negocio – ESB SOA GeHos

  La interacción con los servicios WEB ubicados en el Bus de Servicios se hará utilizando mensajes HL7 sobre XML.

- Middleware – Acceso a datos.

  Se realiza, utilizando las clases agrupadas dentro del Paquete Fábrica de Objetos, estas a su vez hacen referencia a las operaciones de un ayudante (helper) perteneciente a ADO.NET, que se encarga de manejar las conexiones y transacciones a la Base de Datos.

- ESB SOA GeHos – Negocio.

  Las funcionalidades que brinda el negocio, serán reutilizadas por los servicios WEB, mediante la invocación de métodos y las respuestas de estos constituyen colecciones de objetos.

- Acceso a datos – Base de Datos.

  La comunicación se establece a nivel de gestor de bases de datos a través de sus funciones internas.

La arquitectura Cliente/Servidor.

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes. [26]

Entre las principales características de la arquitectura Cliente/Servidor, se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.
- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado.
- Se reduce el tráfico de red considerablemente. Idealmente, el cliente se comunica con el servidor utilizando un protocolo de alto nivel de abstracción.

Modelo Cliente Servidor de tres Capas (Three Tier).

Cada uno de los componentes de la aplicación en una arquitectura de tres capas se separa en una sola entidad. Esto te permite implementar componentes de una manera más flexible, es decir, la aplicación tiene que estar preparada para los posibles cambios que el cliente pueda pedir sin tener que reescribir totalmente la aplicación. Este tipo de arquitectura es la más compleja.

En esta Arquitectura todas las peticiones de los clientes se controlan en la capa correspondiente a la lógica del negocio. Cuando el cliente necesita hacer una petición se la hace a la capa en la que se encuentra la lógica del negocio. Esto es bastante importante pues eso quiere decir que:

- El cliente no tiene que tener drivers ODBC ni la problemática consiguiente de instalación de los drivers por tanto se reduce el costo de mantener las aplicaciones cliente.
- El cliente y el Gestor de Reglas de negocio tienen que hablar el mismo lenguaje.
- El Gestor de Reglas de Negocio y el Servidor de Datos tienen que hablar el mismo lenguaje (ODBC).

Patrones de Diseño.

Los Patrones de Diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

Algunos patrones de diseño escogidos

Fabricación Pura

Patrón que asigna un conjunto altamente cohesivo de responsabilidades a una clase artificial que no representa nada en el dominio del problema: una clase creada para dar soporte a una alta cohesión, un bajo acoplamiento y reutilización.

Para diseñar utilizando este patrón debe buscarse ante todo un gran potencial de reutilización, asegurándose que las responsabilidades de las clases a agrupar en la nueva clase, sean pequeñas y cohesivas, sería lo mismo decir que, estas clases deben tener responsabilidades de *granularidad fina*.

El uso concreto de este patrón se puede evidenciar cuando el producto del negocio del los sistemas, necesitan realizar un proceso, que está compuesto, como proceso, a la vez por procesos independientes, se crean clases que se convierten es una especie de objeto de función central.

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

Fábrica Abstracta

Este patrón proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas.

Se puede usar Fábrica Abstracta cuando: Un sistema debe ser independiente de cómo se crean, componen y representan sus productos, una familia de objetos productos relacionados está diseñada para ser usada conjuntamente, y es necesario hacer cumplir esta restricción o quiere proporcionar una biblioteca de clases de productos y solo quiere revelar sus interfaces, no sus implementaciones.

Dentro de su estructura se encuentran diferentes clases participantes, las cuales se citan a continuación

1) FabricaAbstracta**:** Declara una interfaz para operaciones que crean objetos producto, abstractos.

2) FabricaConcreta: Implementa las operaciones para crear objetos producto concreto.

3) ProductoAbstracto: Declara una interfaz para un tipo de objeto producto.

4) ProductoConcreto: Define un objeto producto para que sea creado por la fábrica correspondiente. Implementa la interfaz ProductoAbstracto.

5) Cliente: Sólo usa interfaces declaradas por las clases FabricaAbstracta y ProductoAbstracto.

Usualmente se crea una única instancia de una clase FabricaConcreta en tiempo de ejecución, quien crea objetos producto que tienen una determinada implementación. Para crear diferentes objetos producto, los clientes deben usar una fábrica concreta diferente.

FabricaAbstracta delega la creación de objetos producto en su subclase FabricaConcreta.

Como parte de una investigación llevada a cabo por los arquitectos del proyecto se definió utilizar el patrón de diseño: Fábrica de Objeto. Se le llama a este patrón de fábrica (Factory en inglés) porque involucran algún tipo de factoría o fabrica de objetos. Los objetos de fabricación tienen la responsabilidad de crear instancias de otras clases. Además tienen la responsabilidad y el conocimiento necesario para encapsular la forma que en que se crean determinados tipos de objetos en una aplicación.

Una fábrica es "una clase que implemente uno o más métodos de creación, que son los métodos que se encargan de crear instancias de objetos (estas instancias pueden ser de esta misma clase o de otras). Esta clase tiene entre sus responsabilidades la creación de instancias de objetos, pero puede tener también otras responsabilidades adicionales. Los métodos de creación pueden ser estáticos

Específicamente existen distintos tipos de patrones de fabricación, tales como: Simple Factory, Factory Method y Abstract Factory.

En este trabajo se hace uso de estos tres tipo el Simple Factory por ejemplo es la clase utilizada para crear instancias de otras clases, Factory Method es que define una interfaz para crear objetos pero deja que sean las subclases las que definan a que clase instanciar; así como el Abstract Factory es que el proporciona una interfaz para crear familias de objetos relacionados y dependientes entre si, sin especificar sus clases concretas.

Mediante una aplicación que se creó en el propio proyecto de de gestión hospitalaria llamada "Create Factory" por un grupo de estudiantes y asesorado por profesores se generaron la capa intermedia o middleware y las entidades del negocio de este módulo de cuerpo de guardia. Esto se hizo gracias al carácter repetitivo que tiene las mismas.

Patrones de Arquitectura

Los patrones de arquitectura escogidos para el desarrollo del sistema son: Arquitectura en Capas y Mapeo de datos.

Arquitectura en Capas

Descompone la aplicación en un conjunto de capas independientes y ordenadas jerárquicamente, cada nivel o capa usa los servicios de la capa inmediatamente inferior y ofrece servicios a la capa inmediatamente superior.

El uso de este patrón brinda la posibilidad de reutilizar un mismo nivel en varias aplicaciones, permite la estandarización, el cambio de nivel no afecta el resto, ahora, no se puede obviar algunos puntos interesantes a la hora de diseñar utilizando este patrón, y estos son: un número excesivo de niveles puede ser ineficiente y un nivel muy bajo hace que las aplicaciones se vuelvan complejas e ineficientes. Si se hace un mal diseño, se tropieza frente a la posibilidad de que cambios de funcionalidad se transmitan de un nivel a otro. [27]

Mapeo de datos

Este patrón propone definir una tabla (si se emplea una bases de datos relacional), para cada clase objeto persistente. Los atributos de los objetos que contienen tipos primitivos de datos (número, cadena, booleano y otros) se mapean en las columnas.

Si un objeto posee atributos exclusivamente de tipos primitivos de datos, el mapeo será simple. Pero la complejidad de la solución puede incrementarse un poco ya que los objetos pueden tener atributos que se refieran a otros objetos complejos, mientras que el modelo relacional exige que los valores sean atómicos.

En este capítulo, se analizan los antecedentes de los sistemas que fueron creados con el propósito de dar solución al caso de estudio que se desarrolla en el trabajo. Además se hace un análisis de las tecnologías y herramientas que serán utilizadas en el desarrollo del sistema propuesto, como son: la plataforma .NET, el lenguaje de desarrollo del lado del servidor, ASP.NET, el lenguaje de desarrollo del lado del cliente JavaScript, la técnica de desarrollo AJAX, la herramienta de desarrollo Visual Studio 2005 y C# como lenguaje base de programación. Además se definen elementos de arquitectura presentes son: Arquitectura en tres capas, Arquitectura Cliente-Servidor, patrones de diseño, tales como, Fábrica Abstracta y Fabricación Pura y los patrones arquitectónicos: Arquitectura en Capas y Mapeo de Objetos.

CAPÍTULO 2. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA.

Este capítulo contiene el resultado de la búsqueda y análisis de la información vinculada al objeto de estudio, procesos a automatizar y conceptos asociados al domino del sistema. Además, se define el diseño de la solución propuesta, los principales elementos tenidos en cuenta a la hora de hacer cada una de las implementaciones para lograr un producto con la calidad y la eficiencia requerida. El desarrollo de este capítulo permitió conocer más el trabajo de los procesos vinculados con las intervenciones quirúrgicas que tienen lugar en los hospitales cubanos por lo que fue necesaria la colaboración de los trabajadores de estos centros. Así como la búsqueda de información referente a dicho tema para lograr que una vez implementado el sistema, este satisfaga las necesidades del usuario final.

2.1 Valoración crítica del diseño propuesto por el analista

El diseño propuesto, se valora como bueno ya que en el se puede obtener las clases fundamentales que deben ser definidas para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, dando una idea clara de lo que se debe implementar. Unido a esto, se encuentran los diagramas de interacción, que explican la colaboración que existe entre las clases y cómo son llamados los métodos y sentencias dentro de cada una, de los cuales se obtiene la información necesaria para conocer el orden de las acciones a implementar. El diseño propuesto fue creado siguiendo patrones, que de manera general constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, basados en la experiencia y que se ha demostrado que funcionan, permitiendo llevar a cabo la implementación clara y limpia del modulo bajo patrones como los GRASP y los GOF.

Los patrones GRASP se utilizan con el objetivo de asignar responsabilidades a las diferentes clases que se definen en el diseño. Dentro de este grupo se identifican cinco patrones muy utilizados: experto, creador, alta cohesión, bajo acoplamiento y el controlador. Estos patrones se les aplican a las clases definidas en el diseño, distribuyendo responsabilidades entre las mismas de forma tal que no existan muchas relaciones, que no se sobrecargue de métodos a una clase en específico pudiendo acomodarlos en otras, entre otras mejoras que brinda el uso de este grupo de patrones.[28]

Los patrones GOF generalmente se evidencian en clases que son creadas debido al uso de un patrón en específico. Existe un grupo de patrones de este tipo definidos para el diseño de clases, con el propósito de crear una arquitectura robusta para el sistema a desarrollar. Del gran número de patrones propuestos por

la pandilla de los cuatro o simplemente GOF, se propone el uso de los patrones Fabricación Pura y Fábrica Abstracta, este último proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas y el anterior asigna un conjunto altamente cohesivo de responsabilidades a una clase artificial que no representa nada en el dominio del problema, una clase creada para dar soporte a una alta cohesión, un bajo acoplamiento y reutilización.[29]

2.2 Análisis de posibles implementaciones, componentes o módulos que puedan ser rehusados. Estrategias de integración

Para dar solución al problema propuesto en el trabajo es necesario hacer uso de otros módulos, como son:

Módulo de Configuración: El cual va ha gestionar toda la información referente a los codificadores necesarios para el funcionamiento del modulo BQG, por ejemplo: los antecedentes patológicos personales, forma de extracción, forma extracción entre otros.

Módulo de Seguridad: El cual va ha gestionar la seguridad y autenticidad del la información. Define el acceso por roles, impidiendo al personal no autorizados entrar el lugares restringidos

Inscripción y Admisión: A través de este módulo podremos realizar las búsquedas de los pacientes, una vez seleccionado el paciente al cual se le realizará la consulta muestra toda la información correspondiente a este paciente. **(Ver anexo 1)**

Laboratorio Clínico: Proporciona la información referente a los resultados de los análisis que se le han indicado al paciente.

Farmacia Permite saber cuáles son los medicamentos que están disponibles en la farmacia, así como los materiales gastables implicados en cada una de las intervenciones quirúrgicas

Para enlazar el Middleware con el Acceso a Datos que esta en el servidor de BD se requiere el uso de ***NpgSQL*.** Interfaz de Programación de Aplicación (API), encargada de la comunicación de una aplicación .Net con servidores de Bases de Datos PostgreSQL. Este Acceso a Datos esta conformado por un conjunto de funciones programadas en lenguaje NpgSql.

Chameleon: Es un subsistema encargado de brindar una interfaz HL7 con la cual interactúan los paquetes de negocio y servicios WEB para lograr el intercambio de información utilizando el estándar HL7.

Dicho subsistema contendrá un conjunto de herramientas que permitirán la administración, integridad y flujo de la información médica entre los distintos Sistemas de Información para la Salud (SIS). Reduce los recursos invertidos en la negociación de las interfaces entre aplicaciones para la salud. Dicha herramienta se encuentra disponible para las plataformas Windows 95/98/2000/XP, Windows Server 2003, Windows NT 3.51 y superior, Mac OS X, Solaris, GNU-Linux, SCO Unix, AS400, Tandem o cualquier sistema operativo que soporte un compilador C++ que cumpla con la norma ANSI 92. Chameleon ha estado en el mercado por más de ocho años y se encuentra desplegado en más de 5000 sitios, además cuenta con una lista de clientes de prestigio internacional que avalan su reputación: Patient Keeper, IBM , AccuImage, Veteran Affairs, University Health Network (UHN), Blue Cross Blue Shield Association, All Meds, + NUTH, LockheedMartin, Aramark, General Electric Medical Systems y SECTRA. Con su utilización se lograría incrementar el prestigio del sistema de salud cubano, así como la confianza internacional en el mismo.

La Capa Intermedia (**Middleware):** Está compuesta por 2 paquetes de clases: Repositorios y Fábrica de Objetos. Para más detalles, ver información contenida en el epígrafe 1.7 Elementos de arquitectura presentes.

Estrategias de integración

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

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

2.3 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. [30]

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. Aunque la legibilidad y la mantenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas.

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y, posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

Las técnicas de codificación incorporan muchos aspectos del desarrollo del software. Aunque generalmente no afectan a la funcionalidad de la aplicación, sí contribuyen a una mejor compresión del código fuente. En esta fase se tienen en cuenta todos los tipos de código fuente, incluidos los lenguajes de programación, de marcado o de consulta.

En general una técnica de codificación no pretende formar un conjunto inflexible de estándares de codificación. Más bien intenta servir de guía en el desarrollo de un estándar de codificación para un proyecto específico de software.

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

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

Para la solución del problema tratado en este trabajo se han utilizado estándares de codificación de la siguiente manera.

Notación Camello

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

Ejemplo:

public GestionarConsulta consulta.

Notación Pascal

En la notación Pascal, si el identificador es simple, el primer carácter se escribe con mayúscula y el resto con minúscula; si el identificador es una palabra compuesta, la segunda palabra debe empezar con mayúscula también.

Es necesario seguir algunas convenciones específicas para los distintos tipos de datos que se mencionan a continuación.

Espacios de nombres (namespaces).

No deben contener espacios y deben definir claramente el conjunto que representan, cada espacio estará separado por punto “.”. Si el identificador del espacio de nombres pequeño (tres o menos caracteres) se escribirá enteramente con mayúscula. Ejemplo:

```
namespace Gehos.Negocio.BQO
{
    //…
}
```

Clases

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

```
public partial class DatosConsulta
{
    //…
}
```

Métodos

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

```
public void BuscarPaciente (DatosPersona persona)
{
    //…
}
```

Propiedades (Properties)

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

```
public String DatosConsulta
{
   //…
}
```

Componentes

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

- TextBoxNombre
- ButtonAceptar
- DropDownListServicio
- GridViewListadoConsulta

Estructura del código.

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

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

Ejemplo:

```
public Int32 IdPersona
{
    get
    {
        return this.idPersona;
    }
    set
    {
```

```
                this. idPersona = value;
            }
        }
```

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

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

Ejemplo:

```
for (int i = 0; i < listadoConsulta.Cont; i++)
{
    if (listadoConsulta[i].Activa )
{
    //….
}
}
```

Como se observa, los estilos de código hacen que el programa fuente sea más legible, lo cual ayuda en la comunicación entre desarrolladores, y permite una temprana detección de errores. La propuesta es completamente extensible.

Siempre existe un riesgo cuando se definen estilos de codificación en aplicar más prefijos o sufijos en la sintaxis a conceptos que ya tienen una forma de ser expresados, cayendo en una redundancia expresiva.

2.4 Descripción de las clases que se utilicen para representar computacionalmente dicha estructura

Estas clases entidad, fueron creadas con el objetivo de agrupar las entidades mapeadas y de crear las relaciones que existen entre ellas. Formando de esta manera las principales entidades del negocio con las que se trabaja. (Tabla 1-13)

Nombre: EPAnestesiaPreoperatoria	
Tipo de clase: Entidad	
Atributo	**Tipo**
anestesiaPreoperatoria	AnestesiaPreoperatoria
anestesiaDosificar	AnestesiaDosificar
Para cada responsabilidad:	
Nombre:	**EPAnestesiaPreoperatoria**
Descripción:	Constructor de la Clase.
Nombre:	**AnestesiaDosificar**
Descripción:	Propiedad para mostrar y modificar la AnestesiaDosificar.
Nombre:	**AnestesiaPreoperatoria**
Descripción:	Propiedad para mostrar y modificar la AnestesiaPreoperatoria.

Tabla 1 Clase Entidad "EPAnestesiaPreoperatoria"

Nombre: EPAnestesiaTransoperatoria	
Tipo de clase: Entidad	
Atributo	**Tipo**
anestesiaTransOperatoria	AnestesiaTransOperatoria
reanimacion	Reanimacion
agenteInduccionTipo	List<AgenteInduccionTipo>
agenteMantenimientoTrans	List<AgenteMantenimientoTrans>
Para cada responsabilidad:	
Nombre:	**EPAnestesiaTransoperatoria**
Descripción:	Constructor de la Clase.
Nombre:	**AnestesiaTransOperatoria**
Descripción:	Propiedad para mostrar y modificar la AnestesiaTransOperatoria.
Nombre:	**Reanimacion**
Descripción:	Propiedad para mostrar y modificar la Reanimacion.
Nombre:	**AgenteInduccionTipo**
Descripción:	Propiedad para mostrar y modificar el AgenteInduccionTipo.
Nombre:	**AgenteMantenimientoTrans**

Descripción:	Propiedad para mostrar y modificar el AgenteMantenimientoTrans.

Tabla 2 Clase Entidad "EPAnestesiaTransoperatoria"

Nombre: EPAnuncioOperatorioGeneral	
Tipo de clase: Entidad	
Atributo	**Tipo**
anuncioOperatorio	AnuncioOperatorio
listSuspensionAnuncio	List<SuspensionAnuncio>
listTipoAnestesiaAnuncio	List<TipoAnestesiaAnuncio>
listTecnicaQuirAnuncio	List<TecnicaQuirAnuncio>
Para cada responsabilidad:	
Nombre:	**EPAnuncioOperatorioGeneral**
Descripción:	Constructor de la Clase.
Nombre:	**TecnicaQuirAnuncio**
Descripción:	Propiedad para mostrar y modificar la TecnicaQuirAnuncio.
Nombre:	**TipoAnestesiaAnuncio**
Descripción:	Propiedad para mostrar y modificar el TipoAnestesiaAnuncio.
Nombre:	**SuspensionAnuncio**

Descripción:	Propiedad para mostrar y modificar la SuspensionAnuncio.
Nombre:	**AnuncioOperatorio**
Descripción:	Propiedad para mostrar y modificar el AnuncioOperatorio.

Tabla 3 Clase Entidad "EPAnuncioOperatorioGeneral"

Nombre: EPConsulta	
Tipo de clase: Entidad	
Atributo	**Tipo**
datosConsulta	DatosConsulta
listAntecedentesPatologicos	List<AntecedentePatologico>
listHabitoToxicoConsulta	List<HabitoToxicoConsulta>
consultaSeguimiento	ConsultaSeguimiento
idConsultaPadre	Int32
epValoracion	EPValoracion
Para cada responsabilidad:	
Nombre:	**EPConsulta**
Descripción:	Constructor de la Clase.
Nombre:	**DatosConsulta**

Descripción:	Propiedad para mostrar y modificar el DatosConsulta.
Nombre:	**AntecedentesPatologicos**
Descripción:	Propiedad para mostrar y modificar el AntecedentesPatologicos.
Nombre:	**HabitoToxicoConsulta**
Descripción:	Propiedad para mostrar y modificar el HabitoToxicoConsulta.
Nombre:	**ConsultaSeguimiento**
Descripción:	Propiedad para mostrar y modificar la ConsultaSeguimiento.
Nombre:	**IdConsultaPadre**
Descripción:	Propiedad para mostrar y modificar el IdConsultaPadre.
Nombre:	**EPValoracion**
Descripción:	Propiedad para mostrar y modificar el EPValoracion.

Tabla 4 Clase Entidad "EPConsulta"

Nombre: EPConsultaAnestesista	
Tipo de clase: Entidad	
Atributo	**Tipo**
anestesiaConsulta	AnestesiaConsulta
listAnestesiaIndicada	List<AnestesiaIndicada>

listPremedicacion	List<Premedicacion>
listAnestesiaExamenFisico	List<AnestesiaExamenFisico>
Para cada responsabilidad:	
Nombre:	**EPConsultaAnestesista**
Descripción:	Constructor de la Clase.
Nombre:	**AnestesiaConsulta**
Descripción:	Propiedad para mostrar y modificar la AnestesiaConsulta.
Nombre:	**AnestesiaIndicada**
Descripción:	Propiedad para mostrar y modificar la AnestesiaIndicada.
Nombre:	**Premedicacion**
Descripción:	Propiedad para mostrar y modificar la Premedicacion.
Nombre:	**AnestesiaExamenFisico**
Descripción:	Propiedad para mostrar y modificar la AnestesiaExamenFisico.

Tabla 5 Clase Entidad "EPConsultaAnestesista"

Nombre: EPConsultaClinico	
Tipo de clase: Entidad	
Atributo	**Tipo**

examenFisico	EPExamenFisico
listInterrogatorio	List<Interrogatorio>
Para cada responsabilidad:	
Nombre:	**EPConsultaClinico**
Descripción:	Constructor de la Clase.
Nombre:	**ExamenFisico**
Descripción:	Propiedad para mostrar y modificar el ExamenFisico.
Nombre:	**Interrogatorio**
Descripción:	Propiedad para mostrar y modificar el Interrogatorio.

Tabla 6 Clase Entidad "EPConsultaClinico"

Nombre: EPConsultaEspecializada	
Tipo de clase: Entidad	
Atributo	**Tipo**
listAnuncioOperatorioGeneral	List<EPAnuncioOperatorioGeneral>
Para cada responsabilidad:	
Nombre:	**EPConsultaEspecializada**
Descripción:	Constructor de la Clase.

Nombre:	**AnuncioOperatorioGeneral**
Descripción:	Propiedad para mostrar y modificar el AnuncioOperatorioGeneral.

Tabla 7 Clase Entidad "EPConsultaEspecializada"

Nombre: EPConsultaPediatra	
Tipo de clase: Entidad	
Atributo	**Tipo**
antecedentesNP	AntecedentesNoPatologicos
listPosnatalesAntecedentes	List<PosnatalesAntecedentes>
listEmbarazoAntecedentes	List<EmbarazoAntecedentes>
Para cada responsabilidad:	
Nombre:	**EPConsultaPediatra**
Descripción:	Constructor de la Clase.
Nombre:	**AntecedentesNoPatologicos**
Descripción:	Propiedad para mostrar y modificar los AntecedentesNoPatologicos.
Nombre:	**PosnatalesAntecedentes**
Descripción:	Propiedad para mostrar y modificar los PosnatalesAntecedentes.
Nombre:	**EmbarazoAntecedentes**

Descripción:	Propiedad para mostrar y modificar el EmbarazoAntecedentes.

Tabla 8 Clase Entidad "EPConsultaPediatra"

Nombre: EPExamenFisico	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenFisico	ExamenFisico
anestesiaExamenFisico	AnestesiaExamenFisico
estertoresExaFisico	List<EstertoresExaFisico>
caractDigestivoExaFisico	List<CaractDigestivoExaFisico>
caractBocaExaFisico	List<CaractBocaExaFisico>
sistNerviosoExaFisico	List<SistNerviosoExaFisico>
Para cada responsabilidad:	
Nombre:	**EPExamenFisico**
Descripción:	Constructor de la Clase.
Nombre:	**ExamenFisico**
Descripción:	Propiedad para mostrar y modificar el ExamenFisico.
Nombre:	**AnestesiaExamenFisico**

Descripción:	Propiedad para mostrar y modificar la AnestesiaExamenFisico.
Nombre:	**EstertoresExaFisico**
Descripción:	Propiedad para mostrar y modificar los EstertoresExaFisico.
Nombre:	**CaractDigestivoExaFisico**
Descripción:	Propiedad para mostrar y modificar el CaractDigestivoExaFisico.
Nombre:	**CaractBocaExaFisico**
Descripción:	Propiedad para mostrar y modificar el CaractBocaExaFisico.
Nombre:	**SistNerviosoExaFisico**
Descripción:	Propiedad para mostrar y modificar el SistNerviosoExaFisico.

Tabla 9 Clase Entidad "EPExamenFisico"

Nombre: EPInformeAnestesia	
Tipo de clase: Entidad	
Atributo	**Tipo**
informeAnestesia	InformeAnestesia
ePAnestesiaPreoperatoria	EPAnestesiaPreoperatoria
ePAnestesiaTransoperatoria	List<EPAnestesiaTransoperatoria>
registroTemporalParametros	List<RegistroTemporalParametros>

anestesiaPosoperatorio	List<AnestesiaPosoperatorio>
Para cada responsabilidad:	
Nombre:	**EPInformeAnestesia**
Descripción:	Constructor de la Clase.
Nombre:	**InformeAnestesia**
Descripción:	Propiedad para mostrar y modificar el InformeAnestesia.
Nombre:	**EPAnestesiaPreoperatoria**
Descripción:	Propiedad para mostrar y modificar la EPAnestesiaPreoperatoria.
Nombre:	**EPAnestesiaTransoperatoria**
Descripción:	Propiedad para mostrar y modificar la EPAnestesiaTransoperatoria.
Nombre:	**RegistroTemporalParametros**
Descripción:	Propiedad para mostrar y modificar el RegistroTemporalParametros.
Nombre:	**AnestesiaPosoperatorio**
Descripción:	Propiedad para mostrar y modificar la AnestesiaPosoperatorio.

Tabla 10 Clase Entidad "EPInformeAnestesia"

Nombre: EPInformeOperatorio
Tipo de clase: Entidad

Atributo	**Tipo**
informeOperatorio	InformeOperatorio
complicaciones	List<Complicaciones>
tecnicaQuirInforme	List<TecnicaQuirInforme>
Para cada responsabilidad:	
Nombre:	**EPInformeOperatorio**
Descripción:	Constructor de la Clase.
Nombre:	**InformeOperatorio**
Descripción:	Propiedad para mostrar y modificar el InformeOperatorio.
Nombre:	**Complicaciones**
Descripción:	Propiedad para mostrar y modificar las Complicaciones.
Nombre:	**TecnicaQuirInforme**
Descripción:	Propiedad para mostrar y modificar la TecnicaQuirInforme.

Tabla 11 Clase Entidad "EPInformeOperatorio"

Nombre: EPPlanificacion	
Tipo de clase: Entidad	
Atributo	**Tipo**

planificacionQuirurgica	PlanificacionQuirurgica
listPlanificacionPorCirujano	List<PlanificacionPorCirujano>
Para cada responsabilidad:	
Nombre:	**EPPlanificacion**
Descripción:	Constructor de la Clase.
Nombre:	**PlanificacionQuirurgica**
Descripción:	Propiedad para mostrar y modificar la PlanificacionQuirurgica.
Nombre:	**PlanificacionPorCirujano**
Descripción:	Propiedad para mostrar y modificar la PlanificacionPorCirujano.

Tabla 12 Clase Entidad "EPPlanificacion"

Nombre: EPValoracion	
Tipo de clase: Entidad	
Atributo	**Tipo**
valoracion	Valoracion
aptoQuirurgico	AptoQuirurgico
interconsulta	Interconsulta
Para cada responsabilidad:	

Nombre:	**EPValoracion**
Descripción:	Constructor de la Clase.
Nombre:	**Interconsulta**
Descripción:	Propiedad para mostrar y modificar la Interconsulta.
Nombre:	**AptoQuirurgico**
Descripción:	Propiedad para mostrar y modificar el AptoQuirurgico.
Nombre:	**Valoracion**
Descripción:	Propiedad para mostrar y modificar la Valoracion.

Tabla 13 Clase Entidad "EPValoracion"

El middleware[2] está compuesta por 2 paquetes de clases: Repositorios y Fábrica de Objetos.

Repositorio para la Consulta Anestesista. (Tabla 14)

Fábrica de Objetos para la Consulta Anestesista. (Tabla 15-17)

Nombre: EPConsultaAnestesistaRepositorio	
Tipo de clase: Controladora	
Atributo	**Tipo**

[2] En el epígrafe 1.7 Elementos de arquitectura presentes, está contenida toda la información referente a esta capa.

factory	IDomainObjectFactory<EPConsultaAnestesista>
Para cada responsabilidad:	
Nombre:	**EPConsultaAnestesistaRepositorio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno***(EPConsultaAnestesista entity)*
Descripción:	Método para obtener una Consulta del Anestesista.
Nombre:	**Adicionar***(IDbTransaction transaction, EPConsultaAnestesista entity)*
Descripción:	Método para agregar una Consulta del Anestesista.
Nombre:	**Actualizar***(IDbTransaction transaction, EPConsultaAnestesista entity)*
Descripción:	Método para modificar una Consulta del Anestesista.

Tabla 14 Clase Entidad "EPConsultaAnestesistaRepositorio"

Nombre: *EPConsultaAnestesistaSelectionFactory*	
Tipo de clase:	
Para cada responsabilidad:	
Nombre:	*EPConsultaAnestesistaSelectionFactory*
Descripción:	Constructor de la Clase.
Nombre:	**Execute***(AdoHelper helper, IDomainObjectFactory<EPConsultaAnestesista> domainObjectFactory, EPConsultaAnestesista entity)*

Descripción:	Devuelve una lista de EPConsultaAnestesista

Tabla 15 Clase Entidad "EPConsultaAnestesistaSelectionFactory"

Nombre: *EPConsultaAnestesistaInsertFactory*	
Tipo de clase:	
Para cada responsabilidad:	
Nombre:	*EPConsultaAnestesistaInsertFactory*
Descripción:	Constructor de la Clase.
Nombre:	**Execute***(AdoHelper helper, IDbTransaction transaction, ref EPConsultaAnestesista entity)*
Descripción:	Inserta en la BD un objeto de tipo *EPConsultaAnestesista.*

Tabla 16 Clase Entidad "EPConsultaAnestesistaInsertFactory"

Nombre: *EPConsultaAnestesistaUpdateFactory*	
Tipo de clase:	
Para cada responsabilidad:	
Nombre:	*EPConsultaAnestesistaUpdateFactory*
Descripción:	Constructor de la Clase.
Nombre:	**Execute***(AdoHelper helper, IDbTransaction transaction, EPConsultaAnestesista entity)*

Descripción:	Actualiza en la BD el objeto de tipo EPConsultaAnestesista.

Tabla 17 Clase Entidad "EPConsultaAnestesistaUpdateFactory"

Nombre: AnuncioOperatorioNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
anuncioOperatorioGeneralRepositorio	EPAnuncioOperatorioGeneralRepositorio
Para cada responsabilidad:	
Nombre:	**AnuncioOperatorioNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno***(EPAnuncioOperatorioGeneral anuncioOperatorioGeneral)*
Descripción:	Método para obtener un Anuncio Operatorio General.
Nombre:	**ObtenerTodos***(EPAnuncioOperatorioGeneral anuncioOperatorioGeneral)*
Descripción:	Método para obtener una lista de los Anuncios Operatorios Generales.
Nombre:	**Adicionar***(EPAnuncioOperatorioGeneral anuncioOperatorioGeneral)*
Descripción:	Método para agregar un Anuncio Operatorio General.
Nombre:	**Actualizar***(EPAnuncioOperatorioGeneral anuncioOperatorioGeneral)*
Descripción:	Método para modificar un Anuncio Operatorio General.

Tabla 18 Clase Controladora "AnuncioOperatorioNegocio"

Nombre: ConsultaAnestesistaNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
consultaAnestesistaRepositorio	EPConsultaAnestesistaRepositorio
Para cada responsabilidad:	
Nombre:	**ConsultaAnestesistaNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno***(EPConsultaAnestesista consultaAnestesista)*
Descripción:	Método para obtener una Consulta del Anestesista.
Nombre:	**Adicionar***(EPConsultaAnestesista consultaAnestesista)*
Descripción:	Método para agregar una Consulta del Anestesista.
Nombre:	**Actualizar***(EPConsultaAnestesista consultaAnestesista)*
Descripción:	Método para modificar una Consulta del Anestesista.

Tabla 19 Clase Controladora "ConsultaAnestesistaNegocio"

Nombre: ConsultaClinicoNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**

consultaClinicoRepositorio	EPConsultaClinicoRepositorio
Para cada responsabilidad:	
Nombre:	**ConsultaClinicoNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno***(EPConsultaClinico consultaClinico)*
Descripción:	Método para obtener una Consulta del Clinico.
Nombre:	**Adicionar***(EPConsultaClinico consultaClinico)*
Descripción:	Método para agregar una Consulta del Clinico.
Nombre:	**Actualizar***(EPConsultaClinico consultaClinico)*
Descripción:	Método para modificar una Consulta del Clinico.

Tabla 20 Clase Controladora "ConsultaClinicoNegocio"

Nombre: ConsultaEspecializadaNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
consultaEspecializadaRepositorio	EPConsultaEspecializadaRepositorio
anuncioOperatorioGeneralRepositorio	EPAnuncioOperatorioGeneralRepositorio
Para cada responsabilidad:	

Nombre:	**ConsultaEspecializadaNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno***(EPConsultaEspecializada consultaEspecializada)*
Descripción:	Método para obtener una Consulta del Especialista.
Nombre:	**Adicionar***(EPConsultaEspecializada consultaEspecializada)*
Descripción:	Método para agregar una Consulta del Especialista.
Nombre:	**Actualizar***(EPConsultaEspecializada consultaEspecializada)*
Descripción:	Método para modificar una Consulta del Especialista.
Nombre:	**ObtenerUno***(EPAnuncioOperatorioGeneral anuncioOperatorioGeneral)*
Descripción:	Método para obtener un Anuncio Operatorio General.
Nombre:	**ObtenerTodos***(EPAnuncioOperatorioGeneral anuncioOperatorioGeneral)*
Descripción:	Método para obtener una lista de los Anuncios Operatorios Generales.
Nombre:	**Adicionar***(EPAnuncioOperatorioGeneral anuncioOperatorioGeneral)*
Descripción:	Método para agregar un Anuncio Operatorio General.
Nombre:	**Actualizar***(EPAnuncioOperatorioGeneral anuncioOperatorioGeneral)*
Descripción:	Método para modificar un Anuncio Operatorio General.

Tabla 21 Clase Controladora "ConsultaEspecializadaNegocio"

Nombre: ConsultaNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
consultaRepositorio	EPConsultaRepositorio
antecedentePatologicoRepositorio	AntecedentePatologicoRepositorio
Para cada responsabilidad:	
Nombre:	**ConsultaNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno***(EPConsulta consulta)*
Descripción:	Método para obtener una Consulta.
Nombre:	**ObtenerTodos***(EPConsulta consulta)*
Descripción:	Método para obtener una lista de Consultas.
Nombre:	**Adicionar***(EPConsulta consulta)*
Descripción:	Método para agregar una Consulta.
Nombre:	**Actualizar***(EPConsulta consulta)*
Descripción:	Método para modificar una Consulta.
Nombre:	**ObtenerTodosAnteriores***(AntecedentePatologico entity)*
Descripción:	Método para obtener una lista de los Antecedentes Patologicos Anteriores.

Nombre:	**ObtenerTodos***(AntecedentePatologico entity)*
Descripción:	Método para obtener una lista de los Antecedentes Patologicos Actuales.

Tabla 22 Clase Controladora "ConsultaNegocio"

Nombre: ConsultaPediatraNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
consultaPediatraRepositorio	EPConsultaPediatraRepositorio
Para cada responsabilidad:	
Nombre:	**ConsultaPediatraNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno***(EPConsultaPediatra consultaPediatra)*
Descripción:	Método para obtener una Consulta del Pediatra.
Nombre:	**Adicionar***(EPConsultaPediatra consultaPediatra)*
Descripción:	Método para agregar una Consulta del Pediatra.
Nombre:	**Actualizar***(EPConsultaPediatra consultaPediatra)*
Descripción:	Método para modificar una Consulta del Pediatra.

Tabla 23 Clase Controladora "ConsultaPediatraNegocio"

Nombre: InformeAnestesiaNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
informeAnestesiaRepositorio	EPInformeAnestesiaRepositorio
Para cada responsabilidad:	
Nombre:	**InformeAnestesiaNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno***(EPInformeAnestesia informeAnestesia)*
Descripción:	Método para obtener un Informe de Anestesia.
Nombre:	**ObtenerTodos***(EPInformeAnestesia informeAnestesia)*
Descripción:	Método para obtener una lista de Informes de Anestesia.
Nombre:	**Adicionar***(EPInformeAnestesia informeAnestesia)*
Descripción:	Método para agregar un Informe de Anestesia.
Nombre:	**Actualizar***(EPInformeAnestesia informeAnestesia)*
Descripción:	Método para modificar un Informe de Anestesia.

Tabla 24 Clase Controladora "InformeAnestesiaNegocio"

Nombre: InformeOperatorioNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
informeOperatorioRepositorio	EPInformeOperatorioRepositorio
Para cada responsabilidad:	
Nombre:	**InformeOperatorioNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno***(EPInformeOperatorio informeOperatorio)*
Descripción:	Método para obtener un Informe Operatorio.
Nombre:	**ObtenerTodos***(EPInformeOperatorio informeOperatorio)*
Descripción:	Método para obtener una lista de Informes Operatorios.
Nombre:	**Adicionar***(EPInformeOperatorio informeOperatorio)*
Descripción:	Método para agregar un Informe Operatorio.
Nombre:	**Actualizar***(EPInformeOperatorio informeOperatorio)*
Descripción:	Método para modificar un Informe Operatorio.

Tabla 25 Clase Controladora "InformeOperatorioNegocio"

Nombre: PlanificacionQuirurgicaNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
planificacionRepositorio	EPPlanificacionRepositorio
Para cada responsabilidad:	
Nombre:	**PlanificacionQuirurgicaNegocio**
Descripción:	Constructor de la Clase.
Nombre:	**ObtenerUno***(EPPlanificacion planificacion)*
Descripción:	Método para obtener una Planificación.
Nombre:	**ObtenerTodos***(EPPlanificacion planificacion)*
Descripción:	Método para obtener una lista de Planificaciones.
Nombre:	**Adicionar***(EPPlanificacion planificacion)*
Descripción:	Método para agregar una Planificación.
Nombre:	**Eliminar***(EPPlanificacion planificacion)*
Descripción:	Método para eliminar una Planificación.
Nombre:	**AdicionarCirujano***(EPPlanificacion planificacion)*
Descripción:	Método para agregar un Cirujano a una Planificación.
Nombre:	**EliminarCirujano***(EPPlanificacion planificacion)*

Descripción:	Método para eliminar un Cirujano de la Planificación.

Tabla 26 Clase Controladora "PlanificacionQuirurgicaNegocio"

Nombre: bqg_ConsultaAnestesia	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSListConsulta	List<EPConsulta>
SSConsulta	EPConsultaAnestesista
ControGuardar	int
SSDatosPersonas	DatosPersonas
ModificarConsulta	Boolean
Para cada responsabilidad:	
Nombre:	**SelecionarConsultaHistorial()**
Descripción:	Método para seleccionar una de las consultas que hay en el historial, se ejecuta en caso que se acceda a la página a través de la búsqueda de consulta o a partir de un anuncio operatorio.
Nombre:	**RefrescarHistorialConsultas()**
Descripción:	Método para mostrar nuevamente la lista de consultas del historial.
Nombre:	**LimpiarConsulta()**

Descripción:	Método para limpiar todos los registros referentes a la consulta. Se ejecuta cuando se selecciona otra consulta.
Nombre:	**ButtonCabecera_Click***(object sender, EventArgs e)*
Descripción:	Evento que se ejecuta al hacer click sobre los botones de la cabecera. Según el botón que fue presionado se ejecutaran acciones como: • Buscar un paciente para una nueva consulta. • Adicionar la consulta actual a la BD • Actualizar la consulta en la BD.
Nombre:	**ControlesGuardados()**
Descripción:	Método para verificar si todos los controles fueron guardados satisfactoriamente.
Nombre:	**gridViewConsultas_RowCreated***(object sender, GridViewRowEventArgs e)*
Descripción:	Evento que se ejecuta cuando el gridview crea cada una de las files. En el se le especifica como se debe crear esta fila.
Nombre:	**gridViewConsultas_PageIndexChanging***(object sender, GridViewPageEventArgs e)*
Descripción:	Evento que se ejecuta al hacer click en el paginado del gridview. En el se le especifica como se debe hacer el paginado.
Nombre:	**gridViewConsultas_SelectedIndexChanging***(object sender, GridViewSelectEventArgs e)*
Descripción:	Evento que se ejecuta al cambiar la selección del gridview. En el se le especifica como se debe hacer la selección del nuevo elemento.
Nombre:	**InavilitarControles()**
Descripción:	Método para inhabilitar los controles cuando haya pasado el tiempo de modificación de la

	consulta.
Nombre:	**multiViewConsulta_ActiveViewChanged***(object sender, EventArgs e)*
Descripción:	Evento que se ejecuta al cambiar de tab. En el se guardan los datos del tab que estaba activo.
Nombre:	**ButtonTab_Click***(object sender, EventArgs e)*
Descripción:	Evento que se ejecuta sobre los botones del tab. En este evento se activa el tab correspondiente al boton pulsado.
Nombre:	**GuardarDatos()**
Descripción:	Método para guardar toda la información recogida durante la consulta en una variable de sesión.

Tabla 27 Clase Interfaz "bqg_ConsultaAnestesia"

Nota: Pantalla asociada en el anexo 4.

Nombre: bqg_ConsultasClinicos	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSListConsulta	List<EPConsulta>
SSConsulta	EPConsultaClinico
ControGuardar	int
SSDatosPersonas	DatosPersonas
ModificarConsulta	Boolean

Para cada responsabilidad:	
Nombre:	**SelecionarConsultaHistorial()**
Descripción:	Método para seleccionar una de las consultas que hay en el historial, se ejecuta en caso que se acceda a la página a través de la búsqueda de consulta o a partir de un anuncio operatorio.
Nombre:	**RefrescarHistorialConsultas()**
Descripción:	Método para mostrar nuevamente la lista de consultas del historial.
Nombre:	**LimpiarConsulta()**
Descripción:	Método para limpiar todos los registros referentes a la consulta. Se ejecuta cuando se selecciona otra consulta.
Nombre:	**ButtonCabecera_Click***(object sender, EventArgs e)*
Descripción:	Evento que se ejecuta al hacer click sobre los botones de la cabecera. Según el botón que fue presionado se ejecutaran acciones como: • Buscar un paciente para una nueva consulta. • Adicionar la consulta actual a la BD • Actualizar la consulta en la BD.
Nombre:	**ControlesGuardados()**
Descripción:	Método para verificar si todos los controles fueron guardados satisfactoriamente.
Nombre:	**gridViewConsultas_RowCreated***(object sender, GridViewRowEventArgs e)*
Descripción:	Evento que se ejecuta cuando el gridview crea cada una de las files. En el se le especifica como debe crear esta fila.

Nombre:	**gridViewConsultas_PageIndexChanging***(object sender, GridViewPageEventArgs e)*
Descripción:	Evento que se ejecuta al hacer click en el paginado del gridview. En el se le especifica como debe hacer el paginado.
Nombre:	**gridViewConsultas_SelectedIndexChanging***(object sender, GridViewSelectEventArgs e)*
Descripción:	Evento que se ejecuta al cambiar la selección del gridview. En el se le especifica como debe hacer la selección del nuevo elemento.
Nombre:	**InavilitarControles()**
Descripción:	Método para inhabilitar los controles cuando haya pasado el tiempo de modificación de la consulta.
Nombre:	**multiViewConsulta_ActiveViewChanged***(object sender, EventArgs e)*
Descripción:	Evento que se ejecuta al cambiar de tab. En el se guardan los datos del tab que estaba activo.
Nombre:	**ButtonTab_Click***(object sender, EventArgs e)*
Descripción:	Evento que se ejecuta sobre los botones del tab. En este evento se activa el tab correspondiente al botón pulsado.
Nombre:	**GuardarDatos()**
Descripción:	Método para guardar toda la información recogida durante la consulta en una variable de sesión.

Tabla 28 Clase Interfaz "bqg_ConsultasClinicos"

Nota: Pantalla asociada en el anexo 2.

Nombre: bqg_ConsultaPediatra	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSListConsulta	List<EPConsulta>
SSConsulta	EPConsultaPediatra
ControGuardar	int
SSDatosPersonas	DatosPersonas
ModificarConsulta	Boolean
Para cada responsabilidad:	
Nombre:	**SelecionarConsultaHistorial()**
Descripción:	Método para seleccionar una de las consultas que hay en el historial, se ejecuta en caso que se acceda a la página a través de la búsqueda de consulta o a partir de un anuncio operatorio.
Nombre:	**RefrescarHistorialConsultas()**
Descripción:	Método para mostrar nuevamente la lista de consultas del historial.
Nombre:	**LimpiarConsulta()**
Descripción:	Método para limpiar todos los registros referentes a la consulta. Se ejecuta cuando se selecciona otra consulta.
Nombre:	**ButtonCabecera_Click***(object sender, EventArgs e)*

Descripción:	Evento que se ejecuta al hacer click sobre los botones de la cabecera. Según el botón que fue presionado se ejecutaran acciones como: • Buscar un paciente para una nueva consulta. • Adicionar la consulta actual a la BD • Actualizar la consulta en la BD.
Nombre:	**ControlesGuardados()**
Descripción:	Método para verificar si todos los controles fueron guardados satisfactoriamente.
Nombre:	**gridViewConsultas_RowCreated***(object sender, GridViewRowEventArgs e)*
Descripción:	Evento que se ejecuta cuando el gridview crea cada una de las files. En el se le especifica como debe crear esta fila.
Nombre:	**gridViewConsultas_PageIndexChanging***(object sender, GridViewPageEventArgs e)*
Descripción:	Evento que se ejecuta al hacer click en el paginado del gridview. En el se le especifica como debe hacer el paginado.
Nombre:	**gridViewConsultas_SelectedIndexChanging***(object sender, GridViewSelectEventArgs e)*
Descripción:	Evento que se ejecuta al cambiar la selección del gridview. En el se le especifica como debe hacer la selección del nuevo elemento.
Nombre:	**InavilitarControles()**
Descripción:	Método para inhabilitar los controles cuando haya pasado el tiempo de modificación de la consulta.
Nombre:	**multiViewConsulta_ActiveViewChanged***(object sender, EventArgs e)*
Descripción:	Evento que se ejecuta al cambiar de tab. En el se guardan los datos del tab que estaba

	activo.
Nombre:	**ButtonTab_Click***(object sender, EventArgs e)*
Descripción:	Evento que se ejecuta sobre los botones del tab. En este evento se activa el tab correspondiente al boton pulsado.
Nombre:	**GuardarDatos()**
Descripción:	Método para guardar toda la información recogida durante la consulta en una variable de sesión.

Tabla 29 Clase Interfaz "bqg_ConsultaPediatra"

Nota: Pantalla asociada en el anexo 3.

Nombre: AnuncioOperatorio	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSListAnuncioHistorial	List<EPAnuncioOperatorioGeneral>
SSConsulta	EPConsultaEspecializada
SSAnuncioOperatorioGeneral	EPAnuncioOperatorioGeneral
SSDatosPersonas	DatosPersonas
PuedeCrearAnuncio	Boolean?
VSSelectedIndex	int
VSPageIndex	int

SSNuevoAnuncio	bool
Para cada responsabilidad:	
Nombre:	**RefrescarHistorialAnuncio()**
Descripción:	Método para mostrar nuevamente la lista de anuncios del historial.
Nombre:	**ButtonTab_Click***(object sender, EventArgs e)*
Descripción:	Evento que se ejecuta sobre los botones del tab. En este evento se activa el tab correspondiente al boton pulsado.
Nombre:	**ButtonCabecera_Click***(object sender, EventArgs e)*
Descripción:	Evento que se ejecuta al hacer click sobre los botones de la cabecera. Según el botón que fue presionado se ejecutaran acciones como: Crear un nuevo anuncio. • Adicionar el anuncio actual a la BD • Actualizar el anuncio actual en la BD. • Ver la consulta a la cual pertenece el anuncio actual.
Nombre:	**NuevoAnuncio()**
Descripción:	Método para crear un nuevo anuncio.
Nombre:	**LimpiarAnuncio()**
Descripción:	Método para limpiar todos los registros referentes al anuncio operatorio. Se ejecuta cuando se selecciona otro anuncio.
Nombre:	**GuardarDatos()**

Descripción:	Método para guardar toda la información recogida durante la consulta en una variable de sesión.
Nombre:	**gridViewAnuncio_SelectedIndexChanging***(object sender, GridViewSelectEventArgs e)*
Descripción:	Evento que se ejecuta al cambiar la selección del GridView. En el se le especifica como debe hacer la selección del nuevo elemento.
Nombre:	**gridViewAnuncio_RowCreated***(object sender, GridViewRowEventArgs e)*
Descripción:	Evento que se ejecuta cuando el GridView crea cada una de las files. En el se le especifica como debe crear esta fila.

Tabla 30 Clase Interfaz "AnuncioOperatorio"

Nota: Pantalla asociada en el anexo 5.

En este capitulo se describe como está implementado el sistema y se realiza una valoración crítica del diseño propuesto por el analista. También se analizan los módulos que se necesitan para funcionar, como son: NpgSQL, Seguridad, Configuración, Chameleon v4.1, Middleware, Inscripción y Admisión, Laboratorio Clínico y Farmacia. Se brinda la descripción de las clases interfaz, controladoras y entidades que conforman la solución propuesta, permitiendo conocer la distribución de las mismas, y facilitando su entendimiento para futuras actualizaciones

CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

3.1 Búsqueda o diseño de los test de unidades que permitan validar la solución propuesta

3.1.1 Niveles de Prueba

Existen diferentes niveles de pruebas, cada una de ellas se realiza en determinados momentos del ciclo de vida del software, la siguiente tabla resume las mismas:

- Unidad.
- Integración.
- Sistema.
- Aceptación.

Este modelo describe a un nivel alto de abstracción las fases del ciclo de desarrollo en las que se involucran las pruebas y los niveles de las mismas, la figura 5 muestra este modelo.

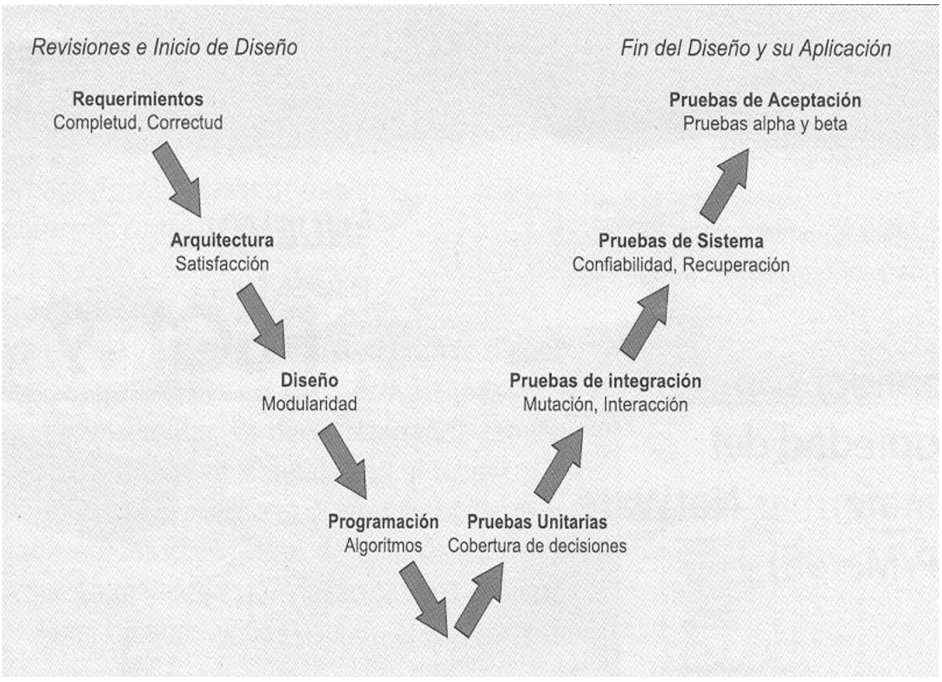


Fig. No. 5 Niveles de prueba con las fases de desarrollo de software

Nivel de Unidad

En este nivel de prueba fundamentalmente se ejecutan casos de pruebas de caja blanca diseñados para:

- Probar las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente, conservan su integridad durante todos los pasos de ejecución del algoritmo.
- Probar las condiciones límites para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento.
- Ejercitar todos los caminos independientes (caminos básicos) de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez. Y, finalmente, se prueban todos los caminos de manejo de errores.

Los casos de pruebas diseñados para ejecutar a nivel de unidad deben descubrir errores como:

- Comparaciones entre tipos de datos distintos.
- Operadores lógicos o de procedencia incorrectos.
- Igualdad esperada cuando los errores de precisión la hacen poco probable.
- Variables o comparaciones incorrectas.
- Terminación de bucles inapropiada o inexistente.
- Fallo de salida cuando se encuentra una iteración divergente.
- Variables de bucles modificadas de forma inapropiada.

La prueba de límites es probablemente la más importante, es una tarea del paso de la prueba de unidad. El software falla con frecuencia en sus condiciones límites. Las pruebas que ejercitan las estructuras de datos, el flujo de control y los valores de los datos por debajo y por encima de los máximos y los mínimos son muy apropiadas para descubrir estos errores. [31]

Nivel de Integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción.

El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño. En el proceso de integración puede derivarse la realización de los casos de uso del sistema, estos describen como interactúan las clases y los objetos y por lo tanto la interacción de los componentes. [32]

Los encargados de diseñar los casos de pruebas de integración consideran como entrada a los diagramas de secuencia de las realizaciones de los casos de uso, pues ahí buscan las combinaciones de entradas, salidas y estado inicial del sistema que dan lugar a escenarios que usan las clases y por lo tanto los componentes que participan en los diagramas. Un caso de prueba se deriva de un diagrama de secuencia, este describe una o varias secuencias del negocio.

Después, se procede a ejecutar el caso de prueba de integración diseñado, creando trazas de ejecución o ejecutándolo paso a paso, a continuación se compara las interacciones actuales con los diagramas de secuencia y de no ser igual se trata de un defecto que puede dar al traste con un error.

Nivel de Sistema

Las pruebas de sistemas caen fuera del ámbito del proceso de software y no las realiza únicamente el desarrollador del software. Sin embargo los pasos durante el diseño del software y durante la prueba pueden mejorar enormemente la posibilidad de de éxito de las pruebas de sistema.

Las pruebas de sistema principalmente se centran en verificar la interacción de los actores con el sistema, por lo que a menudo los casos de pruebas se obtienen a partir de las descripciones de los casos de uso. Aunque también se le aplican prueba al sistema como un todo. [33]

Nivel de Aceptación

La mayoría de los desarrolladores de productos de software llevan a cabo un proceso denominado pruebas Alfa o Beta para descubrir errores considerando que solo el usuario final puede descubrir.

La prueba Alfa es llevada a cabo por el cliente en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador de observador registrando los errores y los problemas de uso. Estas pruebas se llevan a acabo en un entorno controlado.

Las pruebas Betas se llevan a acabo por los usuarios finales en sus puestos de trabajo, y a diferencia de la prueba Alfa el desarrollador no está presente, así que esta prueba no puede ser controlada por el

desarrollador, por lo que el cliente debe registrar todas las inconformidades y tramitárselas al desarrollador. [34]

3.1.2 Métodos de Pruebas

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

La prueba de caja blanca del software se comprueba los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado.

3.1.3 Tipos de Pruebas

Caja blanca

Consiste en realizar pruebas para verificar que líneas específicas de código funcionan tal como está definido. También se le conoce como prueba de caja-transparente. Esta se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar. [35]

Estas tienen como objetivos:

- Garantizar que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejercitar todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecutar todos los bucles en sus límites operacionales.
- Ejercitar las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

Existen varias técnicas para la aplicación de las pruebas de Caja Blanca, en este trabajo se utilizará la técnica del camino básico. Los pasos que se siguen para aplicar esta técnica son:

1. Se numeran las instrucciones del algoritmo y se construye el grafo de flujo. Las instrucciones simples se agrupan en un mismo nodo. Las condiciones compuestas se separan en varios nodos.

2. Calcular la complejidad ciclomática del grafo. Ésta indica el número máximo de caminos linealmente independientes dentro del grafo.

V (G) = Nº de regiones del grafo

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

V (G) = Nº de Nodos Predicado + 1

3. Diseñar los caminos linealmente independientes de menor a mayor de manera que exista como mucho una arista de diferencia entre ellos.

4. Elaborar los casos de prueba (datos de entrada al algoritmo y resultados esperados) para cada uno de los caminos.

Caja negra

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. [35]

La misma no es una alternativa a las técnicas de prueba anteriormente vistas, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca.

Estas pruebas permiten encontrar:

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

- Errores de inicialización y terminación.

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

Para esta prueba, existen varias técnicas, entre ellas están:

1. Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

2. Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar. En este apartado se hará uso de esta técnica para aplicarle a cierto caso de uso del sistema desarrollado el método de la caja negra.

3.2 Descripción de los test de caja negra

Objetivo del test

Validar los siguientes requisitos funcionales:

- Gestionar Hoja de Consulta Anestesia.
- Buscar Anuncio Operatorio
- Buscar Consultas

Alcance

Pruebas de Sistema: Consiste en realizarle pruebas al sistema cuando ya está integrado y consiste en probar los requisitos funcionales definidos por el cliente, en este nivel se realizan principalmente pruebas por el método de prueba de caja negra

Se diseñó y ejecutó casos de pruebas de caja negra con el fin de validar los requisitos funcionales definidos en la etapa final del desarrollo.

Tipo de test y detalles del mismo.

Para este tipo de prueba se utilizó la técnica de valores límites y partición equivalente.

3.3 Evaluación de la ejecución del test y de los resultados obtenidos

Teniendo en cuenta el caso de uso Gestionar Anuncio Operatorio

Las pruebas realizadas a este caso de uso son:

- Crear Anuncio Operatorio.
- Modificar Anuncio Operatorio
- Buscar Anuncio Operatorio

CPR 1: Crear Anuncio Operatorio.

Descripción

El Caso de Uso se inicia cuando el especialista necesita crear, modificar o actualizar el anuncio operatorio. Selecciona la opción de “Gestionar Anuncio Operatorio” en el menú principal de trabajo y habiendo sido localizada esta, llena los datos de la misma, finalizando el caso de uso.

Flujo central

- El sistema muestra la interfaz que permite loguearse al usuario.
- Si el usuario se loguea correctamente con el login especialista y pass especialista el cual tiene permiso de especialista, el sistema muestra la interfaz principal de la aplicación.
- El especialista selecciona la opción “Crear anuncio operatorio” del menú principal de trabajo.
- El especialista llena los datos del anuncio operatorio.

- El especialista selecciona la opción “Ver Planificación Personal”, se ejecuta el caso de uso “Buscar Planificación personal”.
- El especialista completa los datos del anuncio operatorio teniendo en cuenta la planificación quirúrgica.
- El especialista pulsa el botón “Aceptar”

Condiciones de ejecución

- Especialista autentificado satisfactoriamente por el sistema.

Tienen que estar presente en la base de datos

- Registro de personas provenientes del modulo inscripción y admisión.
- Registro de funcionarios provenientes del modulo configuración.
- Servicios.
- Sub servicios.
- Técnicas quirúrgicas.
- Causas de suspensión.

Clases Válidas	Clases no Válidas	Resultados de la prueba.	Resultados de la prueba.	Observaciones	Cumplimiento
El especialista selecciona la opción “Crear anuncio operatorio” del menú principal de trabajo.	No se muestra la interfaz del anuncio operatorio.	Muestra la interfaz del anuncio operatorio.	Satisfactoria	La operación se realizó correctamente	100 %

El especialista selecciona la opción “Ver Planificación Personal”.	El sistema no muestra la planificación personal del especialista.	El sistema muestra la planificación personal del especialista.	Satisfactoria	La operación se realizó correctamente	100 %
El especialista realiza la búsqueda del lente.	El sistema no muestra la interfaz de los lentes.	El sistema muestra la interfaz de los lentes con la refracción esperada para ambas cámara y la disponibilidad.	Satisfactoria	El proceso de mostrado no es de forma instantánea, debido al número de operaciones realizada por el algoritmo para realizar el cálculo de la refracción esperada.	100 %

Tabla 31 “Sección: Crear Anuncio Operatorio”

CPR2: Modificar Anuncio Operatorio

Descripción

El Caso de Uso se inicia cuando el Especialista necesita actualizar (modificar) el anuncio operatorio. Selecciona la opción de “Modificar Anuncio Operatorio” en el menú principal de trabajo y habiendo sido localizada esta, llena los datos de la misma, finalizando el caso de uso.

Flujo central

- El sistema muestra la interfaz que permite loguearse al usuario.
- Si el usuario se loguea correctamente con el login especialista y pass especialista el cual tiene permiso de especialista, el sistema muestra la interfaz principal de la aplicación.
- El especialista selecciona la opción “Modificar anuncio operatorio” del menú principal de trabajo.

- El especialista selecciona la opción “Buscar Paciente” del menú principal de trabajo.
- El especialista modifica los datos del anuncio y pulsa el botón “Aceptar”.

Condiciones de ejecución

- Especialista autentificado satisfactoriamente por el sistema.

Clases Válidas	Clases no Válidas	Resultados de la prueba.	Resultados de la prueba.	Observaciones	Cumplimiento
El especialista selecciona la opción “Modificar anuncio operatorio” del menú principal de trabajo.	No se muestra la interfaz del anuncio operatorio.	Muestra la interfaz del anuncio operatorio.	Satisfactoria	La operación se realizó correctamente	100 %
El especialista selecciona el anuncio operatorio a modificar.	No se muestra el anuncio operatorio deseado.	Muestra el anuncio operatorio deseado.	Satisfactoria	La operación se realizó correctamente	100 %
El especialista modifica los datos del anuncio y pulsa el botón “Aceptar”.	No se registran los datos del anuncio operatorio modificados.	Se registran los datos del anuncio operatorio modificados.	Satisfactoria	Si los datos introducidos no son correctos el sistema muestra una mensaje de error	100 %

Tabla 32 “Sección: Modificar Anuncio Operatorio”

CPR3: Buscar Anuncio Operatorio

Descripción

El Caso de Uso se inicia cuando el usuario desea buscar un anuncio operatorio de un paciente.

Flujo central

- El sistema muestra la interfaz que permite loguearse al usuario.
- Si el usuario se loguea correctamente con el login especialista y pass especialista el cual tiene permiso de especialista, el sistema muestra la interfaz principal de la aplicación.
- El usuario accede al sistema para realizar la búsqueda de un anuncio operatorio.
- El usuario llena los datos necesarios y presiona el botón “Buscar”.
- El usuario accede al anuncio operatorio deseado.

Condiciones de ejecución

- Usuario autentificado satisfactoriamente por el sistema.
- Paciente registrado con anterioridad.

Clases Válidas	**Clases no Válidas**	**Resultados de la prueba.**	**Resultados de la prueba.**	**Observaciones**	**Cumplimiento**
El usuario selecciona la opción “Buscar anuncio operatorio” del menú principal de trabajo.	El sistema no muestra la interfaz correspondiente a la búsqueda de anuncios operatorios.	El sistema muestra la interfaz correspondiente a la búsqueda de anuncios operatorios.	Satisfactoria	El sistema muestra correctamente la interfaz correspondiente a la búsqueda de anuncios operatorios.	100 %

El usuario llena los datos necesarios y presiona el botón "Buscar".	El sistema muestra no muestra ningún anuncio.	El sistema muestra aquellos anuncios que coincidan con los datos entrados.	Satisfactoria	El sistema muestra una lista detallada del o los anuncios que coinciden con los datos entrados.	100 %
El usuario accede al anuncio operatorio deseado.	El sistema no muestra el anuncio operatorio.	El sistema muestra el anuncio operatorio.	Satisfactoria	El sistema muestra el anuncio operatorio haciendo uso de la misma interfaz por la cual fue creado.	100

Tabla 33 "Sección: Buscar Anuncio Operatorio"

Registro de Defectos y Dificultades Detectadas

Elemento	Nº	Descripción de la No conformidad	Aspecto correspondiente	Etapa de la detección	Importante	Recomendaciones
Validación	1	No se validan el código de la unidad, pues entran caracteres extraños como letras.	Esto permite que a la BD entren caracteres no necesarios.	Una Iteración de Prueba	X	Revisar la clase de validaciones.
validación	2	No se validan	Esto permite que a la	Una	X	Revisar la clase de

		el código del área, pues entran caracteres extraños como letras.	BD entren caracteres no necesarios.	Iteración de Prueba		validaciones.
Validación	3	No se validan el código de la unidad, pues entran caracteres extraños como letras.	Esto permite que a la BD entren caracteres no necesarios.	Una iteración de prueba	X	Revisar la clase de validaciones.

Tabla 34 "Sección: "Registro de Defectos y Dificultades Detectadas"

Propuesta de iteración, posibles mejoras a realizar en el diseño.

Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la fase de pruebas. Esta fase del desarrollo de un software es la que mayor cantidad de tiempo y de esfuerzo requiere, se estima que la mitad del esfuerzo de desarrollo de un programa tanto en tiempo como en gastos se invierte en esta. Esta fase añade valor al producto, todos los programas poseen errores y la fase de pruebas los descubre, siendo este el valor que le añade. Mientras más errores se encuentren al software en esta fase mejor. Una prueba del software es un conjunto de actividades que se lleva a cabo sistemáticamente, que puede planificarse por adelantado y ejecutar una vez construido el código para la revisión final de las especificaciones, del diseño y de la codificación del software.

Escala de nivel de gravedad

Detalles: errores de ortografía en pantallas, estética de campos, alineación en impresos, etc.

Errores leves: errores en presentación de datos secundarios, no adecuación a estándares, comportamientos correctos pero diferentes en situaciones similares, dificultades de operación, etc.

Errores comunes: errores en documentos impresos que se entregan a personas ajenas a la organización, errores en presentación de datos, incumplimiento de objetivos en funciones secundarias, caídas de programas auxiliares, etc.

Errores graves: información crítica presentada erróneamente, información mal registrada en la base de datos, caídas de programas, incumplimiento de objetivos en funciones principales, etc.

Errores invalidantes: caídas de programas centrales, corrupción de la base de datos, imposibilidad de operar funciones básicas, errores en imputaciones que afecten cuentas monetarias, etc.

En este capítulo se realiza un estudio de los principales niveles, métodos y tipo de prueba que se llevan a cabo durante el ciclo de vida del software. Se describen los test de caja negra y los valores utilizados, así como, la ejecución de los mismos y los resultados obtenidos. Al caso de uso Gestionar Anuncio Operatorio se le realizaron las siguientes pruebas: Crear Anuncio Operatorio, Modificar Anuncio Operatorio y Buscar Anuncio Operatorio donde no se detectaron errores.

CONCLUCIONES

En el presente trabajo, se llegaron a las siguientes conclusiones:

- Se estudiaron de forma satisfactoria, sistemas informáticos vinculados con la gestión de la información de los bloques quirúrgicos.
- Se realizó un estudio de las principales herramientas y tecnologías más usadas a nivel mundial, lo cual permitió desarrollar la aplicación siguiendo estándares internacionales y las políticas del país en la producción de software.
- Se realizó un estudio de los estándares de codificación, de gran utilidad en la implementación; porque sirvió para aprender nuevos método y formas de tener organizado el código, para en futuros mantenimientos o iteraciones, mejorar la aplicación.
- Se estudió el lenguaje de modelado del análisis, para tener una visión más amplia del mismo, resultando más fácil la comprensión del trabajo para poder avanzar más rápido en la implementación.

  Se ha cumplido el objetivo de la investigación, pues se realizó la implementación del sistema que permite controlar la gestión de la información relacionada con los procesos vinculados con las intervenciones quirúrgicas que tiene lugar en los hospitales del país.

RECOMENDACIONES

Se recomienda a la facultad y a los desarrolladores del área temática de Gestión Hospitalaria:

- Continuar con la implementación del módulo.
- Aplicar los diferentes tipos de pruebas con el objetivo de identificar posibles errores en el mismo.
- Tener en cuenta la propuesta de implementación, para futuros desarrolladores dentro del área temática.
- Valorar la posibilidad de hacer extensiva esta propuesta a otras áreas temáticas relacionadas con la salud.
- La implementación de una versión, para software libre compilada sobre framework Mono, utilizando como herramienta de implementación SharpDevelop y como gestor de base de datos PostgresSQL.

REFERENCIAS BIBLIOGRÁFICAS

[1]. GALA LÓPEZ, BL. SALUD. Proposición de un diseño y premisas teóricas de una historia clínica computarizada para la atención hospitalaria. Rev Cub Informat Med 2002.

[2]. CERRITOS, A.; PUERTO, F. J. F., *et al. Sistema de Información Hospitalaria.* Publicado en el año 2003, última actualización: 25/03/2007. 24 p. Disponible en: www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf.

[3]. Coordinación de informática medica. (2003). Recuperado el 15 de mayo de 2007. Disponible en: http://cim.uag.mx/his.html#nis

[4]. LATORILLA, E. CARE2X - El Proyecto. Publicado en el año 2004, ultima actualización: 08/04/2007, Disponible en: http://www.care2x.org/index.php?c2x_lang=es&chglang=1.

[5]. MÜLLER, P. (1997, Agosto 31). *Globewide Network.* Recuperado el 21 de Abril de 2007. Disponible en: http://www.gnacademy.org/text/cc/Tutorial/Spanish/tutorial.html

[6]. DANIEL, S. G. (2004, Febrero 13). *DEPARTAMENTO DE COMPUTACIÓN*. Recuperado el 5 de Mayo de 2007. Disponible en: https://computacion.cs.cinvestav.mx/~sgarrido/cursos/ing_soft/Componentes/node28.html

[7]. SCHWINDT, A. (7 de Junio de 2006). *Microsoft*. Recuperado el 24 de Mayo de 2007. Disponible en: http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3505.asp

[8]. GARCIA, J. Manual de HTML 2004. Disponible en: http://www.webestilo.com/html/

[9]. PÉREZ, Ivan Nieto. Introducción a JavaScript, 2006.Disponible en: http://www.elcodigo.net/tutoriales/javascript/javascript1.html

[10]. MARTÍNEZ, L. VISUAL BASIC SCRIPT, 2004.Disponible en: http://palmera.pntic.mec.es/~jmart210/personal/aficiones/vbasicscript.htm

[11]. MERELO, J. J. *Transformando documentos XML usando XSLT.* Publicado en el año 2001, última actualización: 25/03/2007. Disponible en: http://geneura.ugr.es/~jmerelo/XSLT/XSLT-2001-1ed.htm.

[12]. ADR INFOR S.L., L. *ASP.NET.* publicado en el año 2005, última actualización: 05/04/2007. Disponible en: http://www.adrformacion.com/cursos/aspnet2/leccion3/tutorial1.html.

[13]. AUTORES, C. D. Manual de PHP 2006. Disponible en: http://es.php.net/manual/es/introduction.php

[14]. MATEU, L. Apuntes de Java 1996. Disponible en: http://www.dcc.uchile.cl/~lmateu/Java/Apuntes/index.htm

[15]. SECO, J. A. G. El lenguaje de programación C#, 2001. Disponible en: http://www.josanguapo.com/

[16]. Ídem [15].

[17]. GARRETT, J. J. *Ajax: Un nuevo acercamiento a las aplicaciones web.* Publicado en el año 2005, última actualización: 10/04/2007. Disponible en: http://www.maestrosdelweb.com/editorial/ajax/.

[18]. SANCHEZ, M. A. M. *Metodologías De Desarrollo De Software.* Publicado en el año 2002, última actualización: 25/02/2007. 5 p. Disponible en: http://www.informatizate.net/articulos/pdfs/metodologias_de_desarrollo_de_software_07062004.pdf.

[19]. Ídem [18].

[20]. Ídem [18].

[21]. Arquitectura básica de la plataforma .Net. Descripción del Framework y sus principales componentes: Lenguajes, biblioteca de clases y CLR., 2006. Disponible en: http://www.desarrolloweb.com/articulos/1328.php.

[22]. AUTORES, C. D. Tutorial 1.4 J2EE. Disponible en: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/

[23]. CÁRDENAS, M. A. C. Mono, lleva el .Net Framework a plataformas no-Windows. Disponible en: http://www.mentores.net/articulos/Mono_Net_Framework_plataformas_noWindows.htm

[24]. Ídem [21].

[25]. Ídem [21].

[26]. 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.

[27]. Ídem [26].

[28]. GUTIERREZ, J. A. S. PATRONES GRASP (Patrones de Software para la asignación General de Responsabilidad), 2007. Disponible en: http://jorgesaavedra.wordpress.com/tag/patrones-grasp

[29]. FACTORY™, D. O. Design Patterns in C# and VB.NET- Gang of Four (GOF) 2005. [25 de Marzo 2007]. Disponible en: http://www.dofactory.com/Patterns/Patterns.aspx

[30]. FERNÁNDEZ, G. Estándar codificación DOTNET, 2005. [8 de Mayo 2007]. Disponible en: http://www.elguille.info/colabora/NET2005/giovannyfernandez_EstandarCodificacionNET.htm

[31]. PRESSMAN, R. Ingeniería del Software: Un enfoque práctico. Madrid: McGraw Hill Prentice-Hall, 2001.

[32]. Ídem [31].

[33]. Ídem [31].

[34]. Ídem [31].

[35]. Ídem [31].

BIBLIOGRAFÍA

1. ADR INFOR S.L., L. ASP.NET, publicado en el año 2005, última actualización: 05/04/2007. Disponible en: http://www.adrformacion.com/cursos/aspnet2/leccion3/tutorial1.html.
2. Daniel, S. G. (2004, Febrero 13). *DEPARTAMENTO DE COMPUTACIÓN*. Recuperado el 5 de Mayo de 2007. Disponible en: https://computacion.cs.cinvestav.mx/~sgarrido/cursos/ing_soft/Componentes/node28.html
3. ASSETA, D. A.; ROMERO, D. D. F., et al. SISTEMAS DE INFORMACION HOSPITALARIA (SIH), Su importancia para el desarrollo de los servicios de salud y el control de la gestión publicado en el año 2004, última actualización: 22/03/2007. 11 p. Disponible en: http://www.medicos-municipales.org.ar/bc1104.htm#1.
4. FERNÁNDEZ, G. Estándar codificación DOTNET, 2005. [8 de Mayo 2007]. Disponible en: http://www.elguille.info/colabora/NET2005/giovannyfernandez_EstandarCodificacionNET.htm
5. Schwindt, A. (7 de Junio de 2006). *Microsoft*. Recuperado el 24 de Mayo de 2007. Disponible en: http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3505.asp
6. CERRITOS, A.; PUERTO, F. J. F., et al. Sistema de Información Hospitalaria, publicado en el año 2003, última actualización: 25/03/2007. 24 p. Disponible en: www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf.
7. MARTÍNEZ, L. VISUAL BASIC SCRIPT, 2004.Disponible en: http://palmera.pntic.mec.es/~jmart210/personal/aficiones/vbasicscript.htm
8. CORPORATION, M. Introducción a Visual Studio, publicado en el año 2007, última actualización: 08/04/2007. Disponible en: http://msdn2.microsoft.com/es-es/library/fx6bk1f4(VS.80).aspx.
9. AUTORES, C. D. Manual de PHP 2006. Disponible en: http://es.php.net/manual/es/introduction.php
10. CORPORATION, M. Usar WSDL, publicado en el año 2007, última actualización: 05/04/2007. Disponible en: http://msdn2.microsoft.com/es-es/library/ms175476(SQL.90).aspx.
11. GARCIA, J. Manual de HTML 2004. Disponible en: http://www.webestilo.com/html/

12. MATEU, L. Apuntes de Java 1996. Disponible en: http://www.dcc.uchile.cl/~lmateu/Java/Apuntes/index.htm

13. AUTORES, C. D. Tutorial 1.4 J2EE. Disponible en: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/

14. KÜNG, S.; ONKEN, L., et al. TortoiseSVN: Un cliente de Subversion para Windows: Version 1.4.3, publicado en el año 2006, última actualización: 08/04/2007. Disponible en: http://ufpr.dl.sourceforge.net/sourceforge/tortoisesvn/TortoiseSVN-1.4.3-es.pdf.

15. Arquitectura básica de la plataforma .Net. Descripción del Framework y sus principales componentes: Lenguajes, biblioteca de clases y CLR., 2006. Disponible en: http://www.desarrolloweb.com/articulos/1328.php.

16. LARMAN, C. UML y patrones. Editado por: Varela, E. F. La Habana: 2004. vol. I y II,

17. LATORILLA, E. CARE2X - El Proyecto, publicado en el año 2004, Disponible en: http://www.care2x.org/index.php?c2x_lang=es&chglang=1.

18. MASADELANTE.COM. ¿Qué es un servidor? - Definición de servidor, publicado en el año 2007, última actualización: 05/03/2007. 1 p. Disponible en: http://www.masadelante.com/faq-servidor.htm.

19. PÉREZ, Ivan Nieto. Introducción a JavaScript, 2006.Disponible en: http://www.elcodigo.net/tutoriales/javascript/javascript1.html

20. MERELO, J. J. Transformando documentos XML usando XSLT, publicado en el año 2001, última actualización: 25/03/2007. Disponible en: http://geneura.ugr.es/~jmerelo/XSLT/XSLT-2001-1ed.htm.

21. GUTIERREZ, J. A. S. PATRONES GRASP (Patrones de Software para la asignación General de Responsabilidad), 2007. Disponible en: http://jorgesaavedra.wordpress.com/tag/patrones-grasp

22. FACTORY™, D. O. Design Patterns in C# and VB.NET- Gang of Four (GOF) 2005. [25 de Marzo 2007]. Disponible en: http://www.dofactory.com/Patterns/Patterns.aspx

23. SECO, J. A. G. El lenguaje de programación C#, 2001. Disponible en: http://www.josanguapo.com/

24. MURIEL, I. M. Lector de Contadores de Agua en Pocket PC bajo .NET Compact Framework. [I+D]. MÁLAGA: UNIVERSIDAD DE MÁLAGA, 2005, [Consultado el: 30/03/2007]. 113 p. Disponible en: http://www.lcc.uma.es/pfc/385.pdf.

25. PRESSMAN, R. Ingeniería del Software: Un enfoque práctico. Madrid: McGraw Hill Prentice-Hall, 2001.

26. PROJECT, W. Historia de la Informática en Cuba, publicado en el año 2007, última actualización: 15/03/2007. 4p. Disponible en: http://es.wikipedia.org/wiki/Historia_de_la_Inform%C3%A1tica_en_Cuba.

27. 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.

28. SANCHEZ, M. A. M. Metodologías De Desarrollo De Software, publicado en el año 2002, última actualización: 25/02/2007. 5 p. Disponible en: http://www.informatizate.net/articulos/pdfs/metodologias_de_desarrollo_de_software_07062004.pdf

29. SPAIN, H. L. ¿Qué es HL7? publicado en el año 2007, última actualización: 07/04/2007. Disponible en: http://www.hl7spain.org/VerPagina.asp?IDPage=0.

30. (W3C. Guía Breve de Servicios Web, publicado en el año 2005, última actualización: 15/03/2007. 5 a. p. Disponible en: http://www.w3c.es/Divulgacion/Guiasbreves/ServiciosWeb.

31. WEBOPEDIA. Browser, publicado en el año 2004, última actualización: 22/03/2007. Disponible en: http://www.webopedia.com/TERM/b/browser.html.

32. WESLEY, A.; RUMBAUGH, E. J., et al. Lenguaje Unificado de Modelamiento, publicado en el año 2000, última actualización: 06/04/2007. Disponible en: http://www.creangel.com/uml/intro.php.

33. WIKIMEDIAPROJECT. XSLT, publicado en el año 2007, última actualización: 06/05/2007. Disponible en: http://es.wikipedia.org/wiki/XSLT.

34. Gala López BL. SALUD. Proposición de un diseño y premisas teóricas de una historia clínica computarizada para la atención hospitalaria. Rev Cub Informat Med 2002.

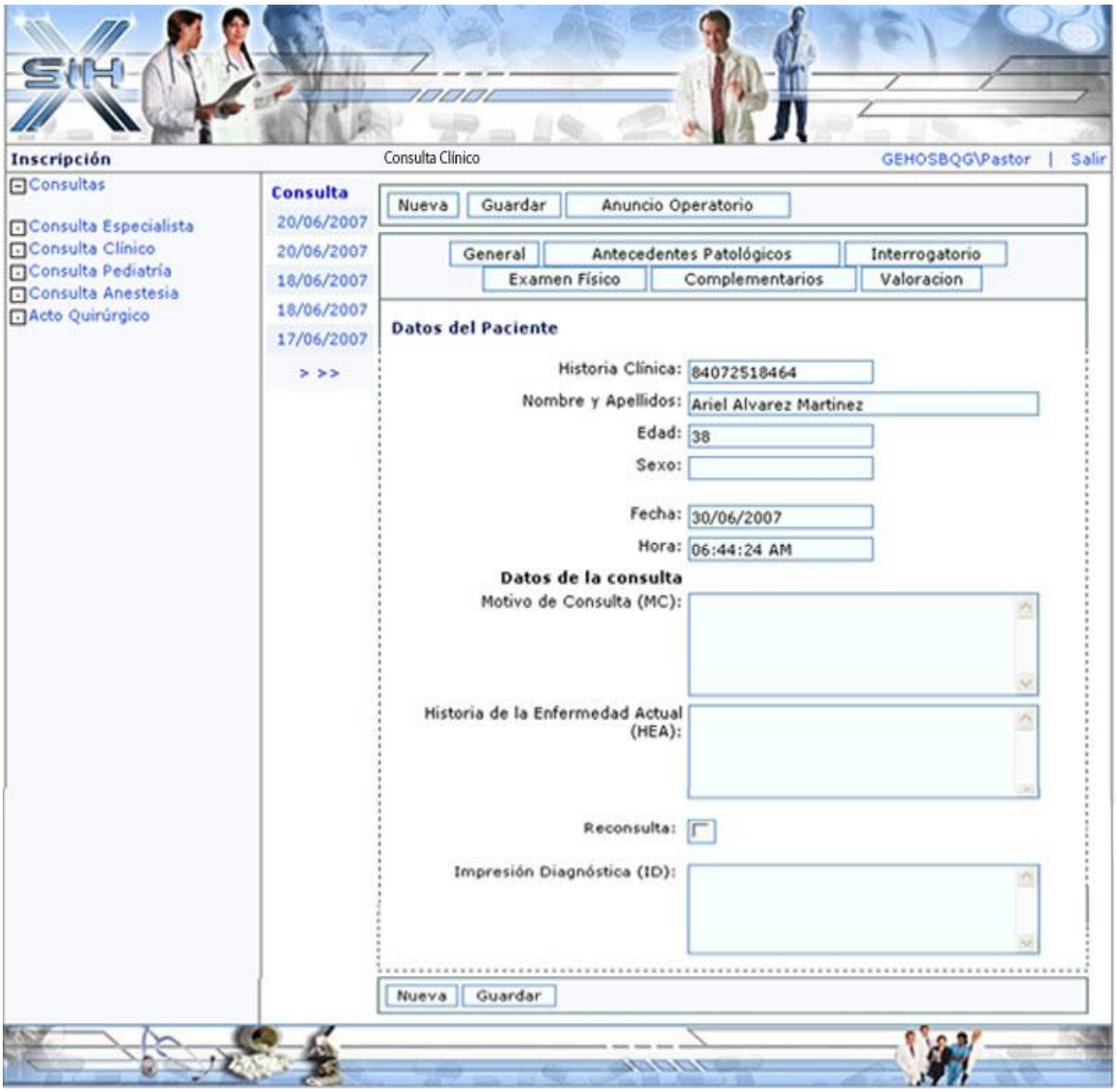
35. Coordinación de informática medica. (2003). Recuperado el 15 de mayo de 2007. Disponible en: http://cim.uag.mx/his.html#nis

36. Müller, P. (1997, Agosto 31). *Globewide Network.* Recuperado el 21 de Abril de 2007. Disponible en: http://www.gnacademy.org/text/cc/Tutorial/Spanish/tutorial.html

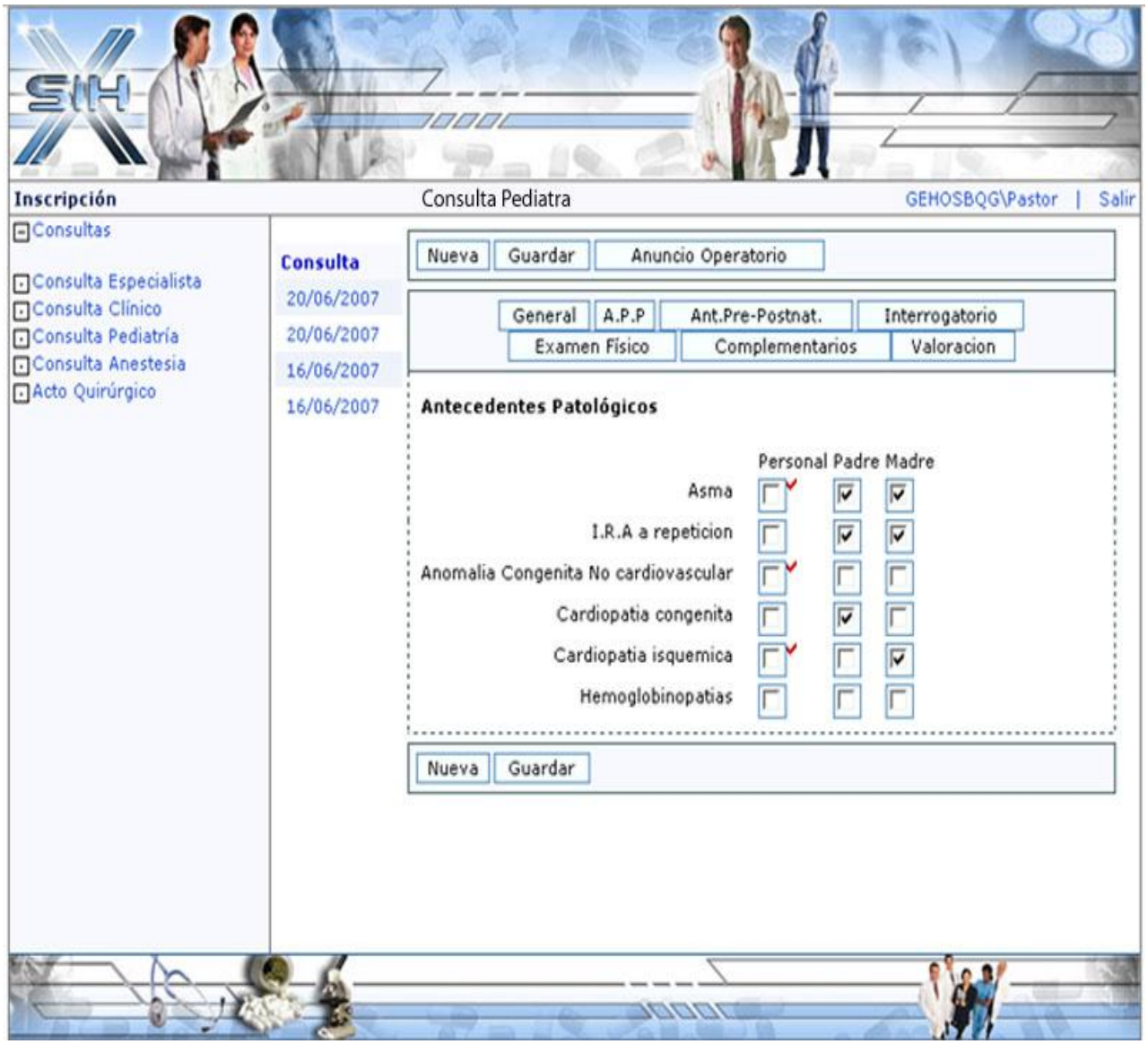
ANEXOS

Anexo No. 1 Interfaz Buscar Paciente.

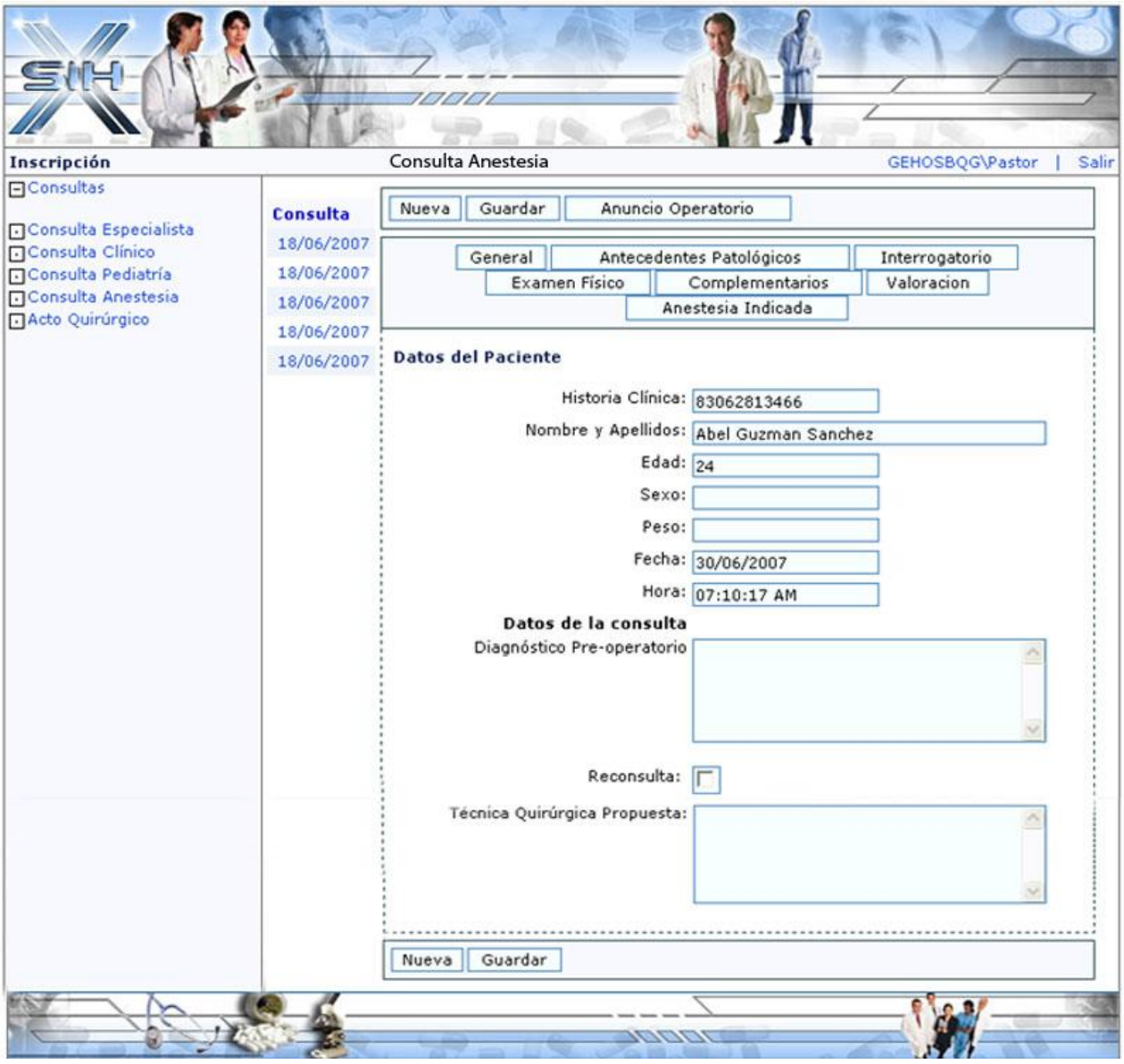
Anexo No. 2 Interfaz Consulta Clínico.



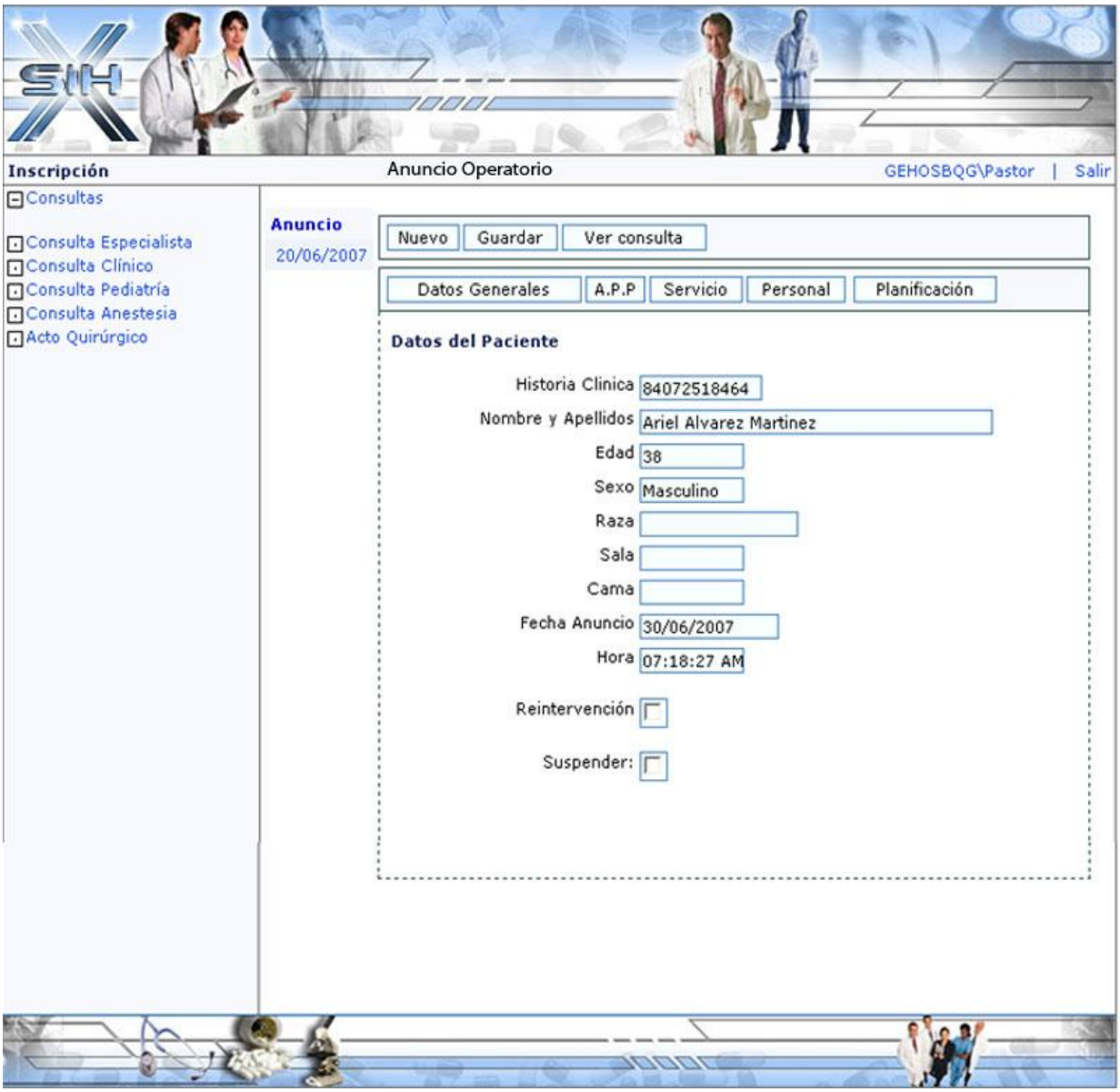
Anexo No. 3 Interfaz Consulta de Pediatría.



Anexo No. 4 Interfaz Consulta del Anestesista.

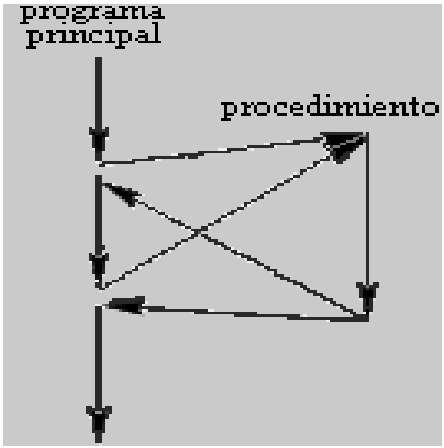


Anexo No. 5 Interfaz Anuncio Operatorio.

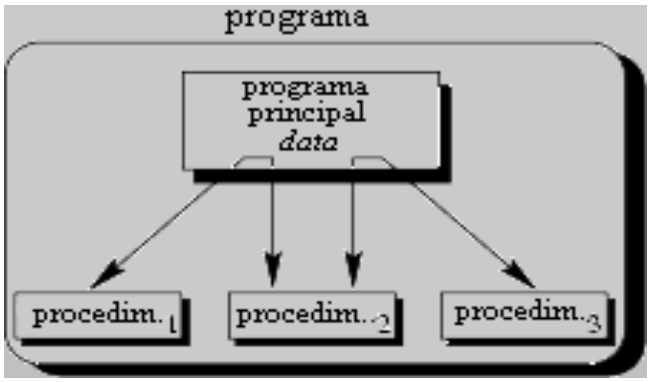


Anexo No.6 Programación no Estructurada. El programa principal opera directamente sobre datos globales.

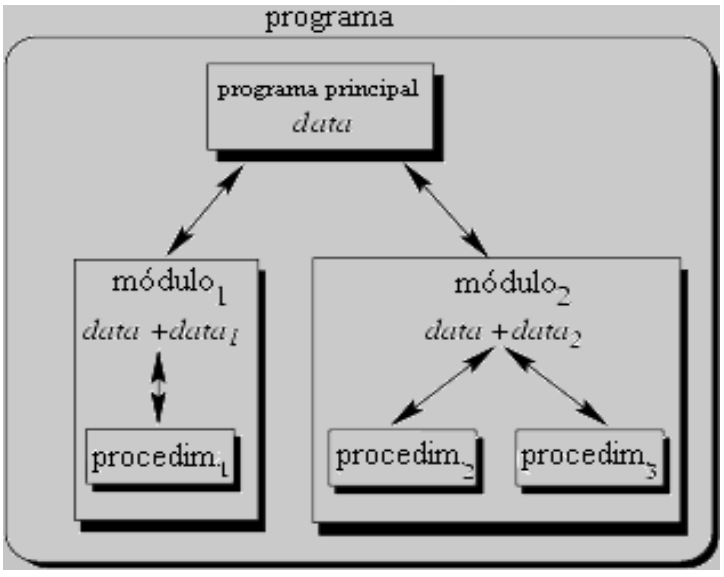
Anexo No. 7 Ejecución de procedimientos. Después del procesamiento, el flujo de controles procede donde la llamada fue hecha.



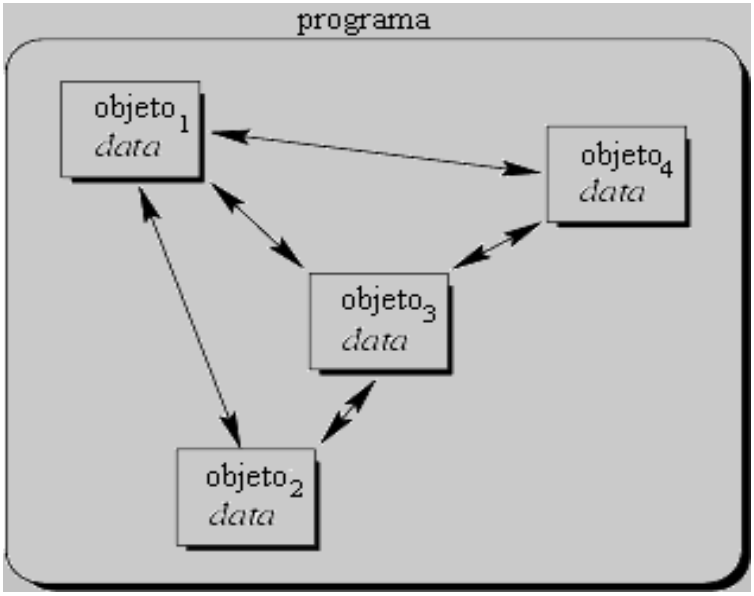
Anexo No. 8 Programación Procedimental. El programa principal coordina las llamadas a procedimientos y pasa los datos apropiados en forma de parámetros.



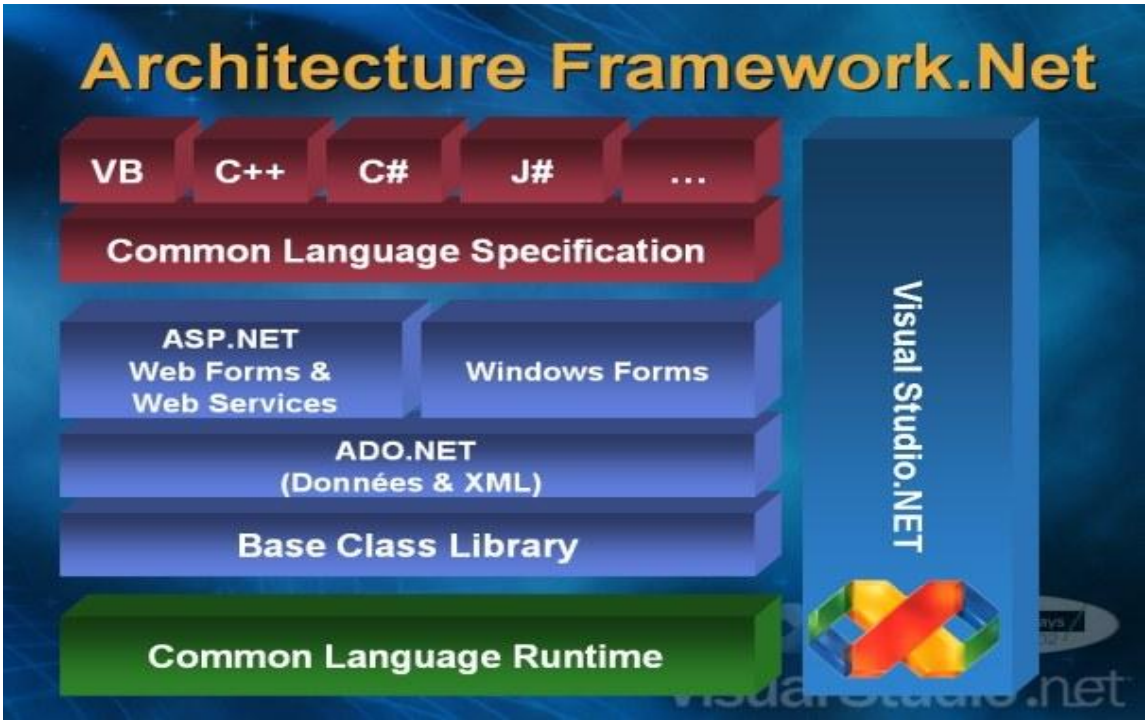
Anexo No. 9 Programación Modular. El programa principal coordina las llamadas a procedimientos en módulos separados y pasa los datos apropiados en forma de parámetros.



Anexo No. 10 Programación Orientada a Objetos. Los objetos del programa interactúan mandando mensajes unos a otros.



Anexo No. 11 Framework.net



GLOSARIO DE TÉRMINOS

FIREWALL: Programa designado para prevenir el acceso no autorizado desde y hacia una red privada.

FTP: Protocolo para la transferencia de archives. Este protocolo es el designado para intercambiar archivos en Internet.

GPL: (*General Public License* o licencia pública general) es una licencia creada por la *Free Software Foundation* a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

GTK#: Grupo de bibliotecas o rutinas para desarrollar interfaces gráficas de usuario (GUI), principalmente para los entornos gráficos GNOME, XFCE y ROX de sistemas Linux.

HTML: Lenguaje de marcado de hipertexto, es el lenguaje autoritario para crear documentos en la World Wide Web. Define la estructura de un documento web usando etiquetas y atributos.

HTTP: Protocolo de transferencia de hipertexto, usado en la World Wide Web. Este protocolo define como los mensajes son formateados y transmitidos, además de cuales acciones deben tomar los servidores web y navegadores en respuesta a varios comandos.

ISO: Organización internacional de estandarización. Ha definido un número importante de estándares computacionales

MSF: El Marco de Soluciones Microsoft abarca temas tales como la estructura del equipo de trabajo que llevará adelante el proyecto (Modelo de Equipos) tanto con el personal de la Consultora como del cliente, y sus roles en el mismo. Cubre el análisis y la mitigación de los riesgos inherentes al proyecto (Modelo de Riesgos), y la forma en que se analiza y establece el alcance del propio proyecto (Modelo de Procesos).

ODBC: Es un estándar de acceso a Bases de Datos desarrollado por *Microsoft Corporation*, el objetivo de *ODBC* es hacer posible acceder a cualquier dato desde cualquier aplicación, sin importar, qué Sistema Gestor de Bases de Datos (*DBMS* por sus siglas en inglés) almacene los datos, *ODBC* logra esto al insertar una capa intermedia llamada manejador de Bases de Datos, entre la aplicación y el *DBMS*, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el *DBMS* entienda.

PostgreSQL: Es un gestor de base de datos muy usado en la actualidad debido a que pertenece a la familia de software libre.

RPC: Protocolo que permite a una computadora ejecutar programas en un servidor.

RUP: Metodología de desarrollo de software basada en UML. Organiza el desarrollo de software en 4 fases.

SGML: Consiste en un sistema para la organización y etiquetado de documentos. La Organización Internacional de Estándares (ISO) ha normalizado este lenguaje en 1986.

SMTP: Protocolo de transferencia de correo simple. Diseñado para enviar mensajes de correo electrónico entre servidores.

SSL: Protocolo desarrollado por Netscape para transmitir datos privados vía internet. Usa un sistema criptográfico compuesto por dos llaves para encriptar los datos.

UDP: Protocolo de conexión. Provee pocos servicios de recuperación de errores aunque ofrece una forma directa para enviar datagramas en una red IP.

XP: Es una disciplina de desarrollo de software que persigue simplificar los procesos de desarrollo. Fue diseñada para ser usada con equipos de desarrollo pequeños que necesiten desarrollos ágiles y con requerimientos cambiantes.