Universidad de las Ciencias Informáticas
Facultad 7

IMPLEMENTACIÓN DE LOS SERVICIOS CIRUGÍA DEL CRISTALINO, CIRUGÍA REFRACTIVA Y PTERIGIUM DEL BLOQUE QUIRÚRGICO OFTALMOLÓGICO Y LOS MÓDULOS DE ADMINISTRACIÓN Y CONFIGURACIÓN

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: Joselín Miló Pérez

Yandy Hernández Carralero

Tutor: Ing. Alexander Rodríguez Rabelo

Ciudad de La Habana, Julio de 2008

¨Año 50 de la Revolución¨

"El único autógrafo digno de un hombre
es el que deja escrito con sus obras."

José Martí.

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 firmamos la presente a los 2 días del mes de julio del año 2008.

Joselín Miló Pérez

_____
Firma del Autor

Yandy Hernández Carralero

_____
Firma del Autor

Alexander Rodríguez Rabelo

_____
Firma del Tutor

DATOS DE CONTACTO

Tutor:
Ing. Alexander Rodríguez Rabelo, profesor graduado de Ingeniero en Ciencias Informáticas en el año 2007 en la Universidad de Ciencias Informáticas.
e-mail: arodriguezra@uci.cu

Agradecimientos

A nuestros padres, por apoyarnos siempre.

A nuestros amigos, por estar a nuestro lado en las buenas y malas.

A los ingenieros Suleydis Suarez y Alexander Rodríguez por su gran apoyo profesional.

A todos los que nos ayudaron de alguna manera en la realización esta investigación.

A la Revolución y en especial al Comandante en Jefe Fidel Castro Ruz por darnos la oportunidad de estudiar en una universidad extraordinaria.

De Joselín:

A mi hermana Yadira por apoyarme siempre.

A mis excelentes amigos desde la infancia Randy, Ariel, Dayron y Pedro Roberto.

A mi gran amigo José Rafael Viera Achang.

Al amigo Antonio (Ñico), por siempre estar ahí cuando lo necesito.

A Leidy y Modesto por todo su apoyo, por haber sido durante estos años como unos segundos padres.

A todas mis nuevos y buenos amigos de la UCI.

A todos muchas gracias por su apoyo.

De Yandy:

A mi hermana por cuidar de mis padres cuando no estoy con ellos.

A mis buenos amigos de la UCI.

Dedicatoria

De Joselín:

A mis padres por ser tan especiales, por ser la razón de mi existir.

A mi hermana por ser tan linda, tan tierna y estar siempre ahí, para mí y para ellos.

A Yule, por ser compañera, amiga, novia, esposa, por ser una de las cosas más lindas que me ha regalado la vida.

A mis abuelitas Carmela y Evelia y a mis dos abuelos en donde quiera que estén.

De Yandy:

A mis padres por su apoyo incondicional, por ser tan especiales.

A mi hermana por quererme tanto.

A mis queridos abuelos.

RESUMEN

Actualmente en los centros oftalmológicos cubanos, no existe una aplicación capaz de gestionar de forma eficiente los grandes volúmenes de información generados por los procesos quirúrgicos; especialmente por la Misión Milagro, causando demoras y perdidas en el proceso de gestión de la información.

Como solución a esta problemática, se realiza el presente trabajo, que tiene como objetivo: implementar un sistema informático que facilite la gestión de la información referente a los servicios cirugía del cristalino, cirugía refractiva y Pterigium, que tiene lugar en los hospitales cubanos. Así como, los módulos de administración y configuración del sistema.

Las principales herramientas y tecnologías utilizadas para la implementación del sistema son la plataforma .NET, ASP.Net para la implementación de las interfaces web, C# como lenguaje del lado del servidor. Además, como lenguaje de desarrollo del lado del cliente se usará JavaScript y la metodología AJAX. También se determinó usar una arquitectura en tres capas. Se utilizó el paradigma de programación orientada a objeto. Usando como IDE de desarrollo Visual Studio 2005.

En el sistema implementado, se utilizan estándares de codificación que permitirán mayor legibilidad y comprensión del código para futuras actualizaciones y mantenimientos para mejorar la aplicación. Con la incorporación de los nuevos servicios (cirugía del cristalino, cirugía refractiva y Pterigium) el sistema implementado será más generalizable; por lo que podrá ser utilizado en los centros oftalmológicos del país, optimizando los servicios ofrecidos a los pacientes y contribuyendo a una mejor organización y menor tiempo de espera por parte de los mismos.

PALABRAS CLAVE

Sistema de Información Hospitalaria, Bloque quirúrgico oftalmológico, Pterigium, Cristalino, Cirugía Refractiva.

TABLAS:

Tabla 1: Clase entidad “Refraccion” 34
Tabla 2: Clase entidad “Queratometria” 35
Tabla 3: Clase entidad “Biometria” 36
Tabla 4: Clase entidad “TensionOcular” 38
Tabla 5: Clase entidad “ExamenRefraccionDinamica” 39
Tabla 6: Clase entidad “ExamenRefraccionIndicada” 41
Tabla 7: Clase entidad “ExamenLensometro” 42
Tabla 8: Clase entidad “PaquimetriaCorneal” 43
Tabla 9: Clase entidad “ExamenesOftalmologicos” 44
Tabla 10: Clase entidad “AntecedentesPatologicosOftalmologicos” 44
Tabla 11: Clase entidad “ExamenCristalino” 45
Tabla 12: Clase entidad “ExamenesAnexos” 46
Tabla 13: Clase entidad “ExamenFondoOjo” 47
Tabla 14: Clase entidad “ExamenMotilidadOcular” 47
Tabla 15: Clase entidad “ExamenReflejoPupilar” 48
Tabla 16: Clase entidad “ExamenSegmentoAnterior” 49
Tabla 17: Clase entidad “ExamenVitreo” 49
Tabla 18: Clase entidad “ParametroSegmentoAnterior” 50
Tabla 19: Clase entidad “AntecedenteOftalmologico” 50
Tabla 20: Clase entidad “HabitoToxico” 51
Tabla 21: Clase entidad “Otropia” 52
Tabla 22 Clase entidad “Patologia” 52
Tabla 23: Clase entidad “EPRefraccion” 53
Tabla 24: Clase entidad “EPQueratometria” 54
Tabla 25: Clase entidad “EPBiometria” 54
Tabla 26: Clase entidad “EPTensionOcular” 55
Tabla 27: Clase entidad “EPLensometro” 55
Tabla 28: Clase entidad “EPPaquimetriaCorneal” 56
Tabla 29: Clase entidad “EPRefraccionIndicada” 57
Tabla 30: Clase entidad “EPRefraccionDinamica” 57
Tabla 31: Clase controladora “RefraccionRepositorio” 58
Tabla 32: Clase controladora “RefraccionSelectionFactory” 59
Tabla 33: Clase controladora “RefraccionInsertFactory” 59
Tabla 34: Clase controladora “RefraccionUpdateFactory” 60
Tabla 35: Clase controladora “RefraccionDeleteFactory” 60
Tabla 36: Clase controladora “EPRefraccionNegocio” 61
Tabla 37: Clase controladora “EPQueratometriaNegocio” 62
Tabla 38: Clase controladora “EPBiometriaNegocio” 63
Tabla 39: Clase controladora “EPTensionOcularNegocio”. 64
Tabla 40: Clase controladora “EPLensometroNegocio”. 65
Tabla 41: Clase controladora “EPRefraccionIndicadaNegocio”. 66
Tabla 42: Clase controladora “EPRefraccionIndicadaNegocio”. 67
Tabla 43: Clase controladora “EPPaquimetriaCornealNegocio”. 68
Tabla 44: Clase interfaz “bqo_CirugiaRefractiva” 71

Tabla 45: Clase interfaz “bqo_CirugiaCristalino” 74
Tabla 46: Clase interfaz “bqo_CirugiaPterigium” 76
Tabla 47: Clase interfaz “ExamenEspCristalino” 78
Tabla 48: Clase interfaz “ExamenEspRefractiva” 80
Tabla 49: Clase interfaz “GestionarUsuario” 83
Tabla 50: Clase interfaz “AgregarUsuario” 86
Tabla 51: Clase interfaz “AgregarMetodo” 87
Tabla 52: “Sección: Insertar nuevo examen especializado” 103
Tabla 53: “Sección: Consultar examen especializado” 103

Tabla de Contenidos

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 5

1.1 CONCEPTOS BÁSICOS RELACIONADOS CON EL DOMINIO DEL PROBLEMA. 5

1.2 SISTEMAS AUTOMATIZADOS EXISTENTES VINCULADOS AL CAMPO DE ACCIÓN. 5

1.3 ANÁLISIS DE LAS POSIBLES METODOLOGÍAS, TÉCNICAS Y HERRAMIENTAS DE DESARROLLO A UTILIZAR. 8

1.3.1 Arquitectura Cliente – Servidor 8

1.3.2 Aplicaciones WEB 8

1.3.3 Navegadores Web 9

1.3.4 Técnicas de programación. 9

1.3.5 Lenguajes y plataformas de desarrollo. 11

1.3.6 Lenguaje del lado del servidor. 12

1.3.7 Tecnologías de desarrollo 13

1.3.8 Plataformas 15

1.3.9 Entorno de desarrollo integrado 17

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA 20

2.1 VALORACIÓN CRÍTICA DEL DISEÑO PROPUESTO POR EL ANALISTA. 20

2.2 ANÁLISIS DE POSIBLES IMPLEMENTACIONES, COMPONENTES O MÓDULOS YA EXISTENTES QUE PUEDAN SER REHUSADOS. 21

2.3 ESTRATEGIAS DE INTEGRACIÓN 22

2.4 DESCRIPCIÓN DE LOS ALGORITMOS NO TRIVIALES A IMPLEMENTAR. 23

2.4.1 Análisis de complejidad. 23

2.5 ESTÁNDARES DE CODIFICACIÓN 27

2.5.1-Notación Camello. 28

2.5.2-Notación Pascal 29

2.6 DESCRIPCIÓN DE LAS NUEVAS CLASES U OPERACIONES NECESARIAS. 33

2.6.1 Entidades del negocio 33

2.6.2 Clases Controladoras. 58

2.6.3 Clases interfaces. 69

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA 89

3.1 PRUEBAS A REALIZAR EN TIEMPO DE DESARROLLO. 90

3.2 PRUEBAS DESPUÉS DE LA PROGRAMACIÓN. 90

3.3 PRUEBAS DE CAJA BLANCA. 92

3.4 PRUEBAS DE CAJA NEGRA. 99

CONCLUSIONES 105

RECOMENDACIONES 106

REFERENCIAS BIBLIOGRÁFICAS 107

BIBLIOGRAFÍA 109

INTRODUCCIÓN

En la actualidad el estado cubano se encuentra inmerso en lo que se ha llamado la “Informatización de la Sociedad Cubana”, proyecto mediante el cual se aplican las Nuevas Tecnologías de la Informatización y las Comunicaciones (NTIC) a las diferentes esferas y sectores de la sociedad para lograr como resultado una mayor productividad, eficiencia y eficacia con la optimización de los recursos.

Es imposible tratar este tema, sin hacer mención al sector de la salud, pues en este sector por las implicaciones humanas y sociales que tiene para con el pueblo, es relevante la aplicación de estas tecnologías, especialmente para la parte asistencial, donde está presente el diagnóstico, tratamiento, rehabilitación, farmacología, laboratorio clínico, oftalmología, entre otros.

El uso de la Informática en la Medicina es una de las aplicaciones más comunes e importantes desde hace varias décadas, y ha permitido al sector de la salud, no sólo contar con métodos novedosos, sencillos y eficaces de gestión administrativa en consultas, hospitales y centros de investigación biomédica, sino también disponer de *software* que reducen la posibilidad de error en el diagnóstico de las enfermedades, y que aceleran su formulación. [1]

El sentido de la visión, uno de los más importantes en la evolución del ser humano, recibe especial atención en el sistema cubano de salud, con los procedimientos más avanzados para el enfrentamiento a diversas dolencias.

Los servicios cubanos de salud ofrecen la posibilidad de acceder a tratamientos para los más diversos trastornos de la visión, entre los cuales se incluyen la catarata, glaucoma, Pterigium, cristalino, miopía y trasplante de córnea.

En esa materia, Cuba y Venezuela promueven la iniciativa de salud conocida como Operación Milagro, la cual está dirigida a ofrecer alternativas de cura a miles de personas aquejadas de los más diversos males de la vista, todo ello con el empleo de la más moderna tecnología.

La prevalencia de la ceguera entre los ciudadanos varía en relación al grado de desarrollo económico de cada país. Mientras que para los países desarrollados es de un 0.25%, en países con economía y servicios de salud muy pobres llega a alcanzar el 1% de la población.[2]

Esta Misión Milagro viene realizándose desde hace algunos años, generando grandes volúmenes de información sobre los pacientes atendidos, por lo que se debe tener un alto nivel de control de esta información para lograr un mejor desempeño del proceso médico y de los recursos que se emplean.

INTRODUCCIÓN

El Instituto Cubano de Oftalmología “Ramón Pando Ferrer” forma parte principal de este sistema en nuestro país, por lo que su informatización es imprescindible en el proceso de informatización de la salud cubana.

A finales de la década del 80 en dicho hospital se desarrolló un sistema que permite gestionar y administrar parte de la información generada durante los procesos por los que atraviesa un paciente al ser atendido, el Sistema Automatizado de Microcirugía (SAMC), el cual, en la actualidad se encuentra implantado y en funcionamiento.

Dicho sistema está implementado en "Clipper'85 Winter", lenguaje de programación creado en 1985 por Nantucket Corporation y herramienta líder de desarrollo, bajo sistema operativo MS-DOS, de aplicaciones relacionadas con bases de datos, sobre todo programas de facturación y contabilidad.

Debido a los cambios realizados en este centro oftalmológico en los últimos años y los grandes volúmenes de información que se genera sobre los pacientes atendidos y que dicho sistema no gestiona todos los procesos que se desarrollan actualmente en el mismo, este ha dejado de responder a los intereses principales de los usuarios.

Para que todos los procesos que se desarrollan puedan llegar a buen fin, tanto su intervención en el ámbito de la salud como en el plano social requieren de una enorme capacidad de planificación, ejecución y control.

En este centro hospitalario se gestiona gran cantidad de información referente a los servicios quirúrgicos que se realizan, los medios utilizados para el manejo de esta, actualmente son de forma manual lo que conlleva a demora en la recepción de los datos y el nivel de error en estos sea considerable, así como la planificación de las consultas para estos servicios.

En estos momentos se trabaja integradamente en el desarrollo de un grupo de aplicaciones básicas para la informatización del sector de la salud. Con el desarrollo tecnológico existente, es posible realizar un proceso de informatización en los hospitales cubanos que permita gestionar: buscar, obtener, actualizar, mostrar, recuperar, eliminar o modificar los datos de manera clara, sencilla, ordenada y segura.

Mediante los métodos empleados actualmente el trabajo de manipulación y almacenamiento de los datos es lento y en ocasiones se necesitan con rapidez por lo que sería factible obtener con gran velocidad la información en muy poco tiempo.

Basado en lo antes expuesto se tiene como **problema científico:** ¿Cómo facilitar la gestión de la información referente a los servicios quirúrgicos de la especialidad de oftalmología en los hospitales cubanos?

El **objeto de estudio** lo constituyen los procesos de gestión de la información hospitalaria en la especialidad de Oftalmología.

El **campo de acción** está centrado en el proceso de gestión de la información relacionada con las intervenciones quirúrgicas oftalmológicas que tiene lugar en los hospitales cubanos.

Como **objetivo general** se define: Implementar un sistema informático que facilite la gestión de la información referente a los servicios cirugía del cristalino, cirugía refractiva y Pterigium, que tiene lugar en los hospitales cubanos. Así como los módulos de administración y configuración.

Para lograr el cumplimiento de los objetivos se proponen las **tareas** siguientes:

- Estudio crítico del análisis realizado sobre el funcionamiento del Bloque Quirúrgico Oftalmológico en los hospitales.
- Hacer un análisis crítico de las tecnologías y lenguaje a utilizar.
- Estudio de las herramientas y metodologías a utilizar.
- Utilizar en la implementación los Patrones de Diseño definidos en el análisis.
- Implementar la capa de acceso a datos del Módulo.
- Implementar las funcionalidades de los servicios oftalmológicos.
- Implementar las funcionalidades de los módulos de Administración y Configuración del Sistema.
- Brindar una interfaz gráfica orientada al usuario y a las exigencias del mercado internacional.
- Analizar la estrategia de integración de la solución con otros módulos o sistemas.
- Realizar las pruebas necesarias para garantizar el correcto funcionamiento de la aplicación implementada.
- Realizar una valoración de los resultados.

La estructuración del contenido del presente trabajo es la siguiente:

El Capítulo I titulado “Fundamentación teórica”: ofrece los conceptos básicos asociados al negocio y los métodos científicos que se utilizaron. Se brinda el estado del arte en cuanto a técnicas, tecnologías, metodologías y *software* utilizados en la actualidad o en las que se apoya para la solución del problema que se enfrenta.

El Capítulo II denominado “Descripción y análisis de la solución propuesta”: se plantea una valoración crítica del diseño propuesto por el analista, análisis de posibles implementaciones de componentes o módulos ya existentes que puedan ser rehusados y las estrategias de integración, una descripción de los algoritmos no triviales a implementar. Análisis de complejidad de los mismos y selección de las estructuras de datos apropiadas para la implementación de estos algoritmos, finalizando con la descripción de las nuevas clases u operaciones necesarias.

En el Capítulo III “Validación de la solución propuesta”: se muestran las pruebas realizadas para validar la solución propuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se realiza un análisis detallado del estado del arte de los distintos sistemas especializados en oftalmología. Así como, las distintas técnicas de programación que existen a nivel internacional, nacional y en la universidad. De las tendencias, técnicas, tecnologías, metodologías relacionadas con dichas técnicas y plataformas de desarrollo que la soportan.

1.1 Conceptos básicos relacionados con el dominio del problema.

Existen una serie de conceptos necesarios para la posterior comprensión del presente, a continuación se exponen los mismos.

La ***Oftalmología*** es la especialidad médico-quirúrgica que se relaciona con el diagnóstico y tratamiento de los defectos y de las enfermedades del aparato de la visión. [3]

Bloque Quirúrgico Oftalmológico**:** Es el área centralizada en la que se genera toda la actividad quirúrgica del hospital oftalmológico.

La ***cirugía*** es el tratado de las enfermedades que se pueden curar con procedimientos manuales empleados según cada enfermo.

Pterigium es una enfermedad que afecta la conjuntiva y la cornea. Este crece en forma de una masa carnosa sobre la cornea - la estructura transparente localizada frente al iris (el que da color al ojo).

La ***cirugía refractiva*** consiste en una serie de procedimientos quirúrgicos, compuestos por técnicas, que permiten que los pacientes que utilizan en forma permanente anteojos, o lentes de contacto, dejen de hacerlo inmediatamente. Estas técnicas apuntan a resolver anomalías refractivas que afectan a cada paciente en particular.

La ***cirugía del cristalino*** es un procedimiento para retirar del ojo el cristalino que se ha vuelto opaco (catarata) con el fin de mejorar la visión.

1.2 Sistemas automatizados existentes vinculados al campo de acción.

Es importante antes de comenzar el desarrollo de cualquier aplicación el estudio de los antecedentes relacionados con el campo de acción que concierne a la misma, para obtener de aquí, una línea base en cuanto a la estructura y las principales funcionalidades.

En la actualidad existen en el mundo múltiples sistemas para la gestión oftalmológica entre los que se destacan:

Oftalmosalus

Oftalmosalus es un *software* propietario especializado para la Gestión de Clínicas y Consultas de Oftalmología. El *software*, desarrollado por QSOFT desde 1995, permite gestionar de manera completa e integrada las 3 grandes áreas de Gestión del Centro: Agendas, Historias Clínicas y Facturación.

OFTALMOSALUS está actualmente implantado en Centros Oftalmológicos de referencia en España y otros países.[4]

Medisys Oftal

Las primeras instalaciones de Medisys Oftal para Windows en España se realizaron en el año 1996.

Sus principales funcionalidades son:

- Administración de datos del paciente.
- Consultorio electrónico.
- Sistema de captura de imágenes.
- Visor de imágenes.
- Procesamiento de textos integrado.
- Recetas Listas de pacientes y sus valores de graduación para Lasik.

Los programas de Medisys son aplicaciones para: Windows 98, Windows 2000 y Windows XP.[5]

Software Modular para consultorios médicos

Historia Clínica interactiva modular e integradora de especialidades médicas, diseñada para el especialista y/o médico generalista, en la tarea del seguimiento transversal y longitudinal del paciente, en las áreas de mayor interés de cada especialidad.

Se trata de un Asistente de Consultorio, desarrollado por equipos formados por médicos especialistas y profesionales informáticos, destinado a centros de consultorios, clínicas, sanatorios y hospitales, pudiendo ser utilizado simultáneamente, por distintos profesionales, conectados en red entre ellos y la recepción.

Consta con un Módulo de Oftalmología que brinda las siguientes funcionalidades: [6]

- Protocolo Gráfico e interactivo de la Consulta Oftalmológica.
- Extensa base de datos de Signos y Síntomas Oftalmológicos.
- Registro Digital de Estudios c/aparatos especializados.
- Herramienta Gráfica para Registro de Estrabismo.
- Herramienta Gráfica para Registro de Fondo de Ojo.
- Herramienta Gráfica para registro de Biomicroscopía.
- Base de datos de Tratamientos Oftalmológicos.
- Panel de Alarmas Inteligentes y Manuales.
- Recetario de Anteojos.
- Material Instructivo para el paciente.

VisionDat Software

Es un programa de tecnología avanzada que ofrece diversas opciones para el control y manejo de datos en consultorios.

Este sistema permite tener control sobre: [7]

- Archivo de pacientes y su evolución por fechas.
- Registro de diagnósticos de segmentos interno y externo.
- Diagnóstico refractivo, de visión binocular y anexos.
- Elaboración de recetas y recomendaciones a su paciente.
- Control de laboratorios de lentes de aro y de contacto.
- Control de citas y actividades.
- Estadísticas totales de sus registros.
- Estadísticas particulares de sus pacientes, tales como evolución de poder refractivo y gráficas de eficiencia y pérdida visual.

Actualmente solo existe un software oftalmológico funcionando en nuestro país.

S.A.M.C (Sistema Automatizado de Microcirugía.)

Sistema que está funcionando actualmente en el hospital oftalmológico "Ramón Pando Ferrer", capaz de gestionar parte de la información generada durante el proceso quirúrgico por el que transita un paciente. Aun dista de gestionar todos los procesos que en este se gestionan. Desarrollado en Clípper sobre MS-DOS y bases de datos ficheros mdf.

Del análisis de estos sistemas se ha obtenido la línea base de desarrollo del presente trabajo, además de valorar los antecedentes existentes para lograr dar cumplimiento a los requisitos establecidos por el cliente de manera homogénea con los *softwares* existentes en el mundo.

1.3 Análisis de las posibles metodologías, técnicas y herramientas de desarrollo a utilizar.

Para alcanzar un nivel técnico en el desarrollo de un software, acorde con el desarrollo actual en la automatización de la información, se hace indispensable el estudio detallado de las tecnologías a utilizar, precisando de cada una, las ventajas y desventajas que representan para alcanzar y mantener el nivel técnico esperado. A continuación, se describe las principales tecnologías, conceptos, herramientas y propuestas para el desarrollo de la solución tratada en el presente sistema.

1.3.1 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.

1.3.2 Aplicaciones WEB

Una aplicación Web es un conjunto de páginas Web estáticas y dinámicas. Sistema informático que los usuarios utilizan accediendo a un servidor web a través de Internet o de una intranet.

Una página Web estática es aquélla que no cambia cuando un usuario la solicita: el servidor Web envía la página al navegador solicitante sin modificarla. Por el contrario, el servidor modifica las

páginas Web dinámicas antes de enviarlas al navegador solicitante. La naturaleza cambiante de este tipo de página es la que le da el nombre de dinámica.

1.3.3 Navegadores Web

Un navegador web (en inglés, browser) es un programa informático (software) que permite visualizar y navegar (de ahí su nombre) por las páginas web. Las páginas web están compuestas por diferentes elementos: texto, imágenes, listas y otros. El navegador web identifica todos los elementos que conforman la página para generar su representación visual. En la representación aparecerán los enlaces. El navegador web permitirá el utilizar los enlaces para navegar entre las páginas.[8]

1.3.4 Técnicas de programación.

1.3.4.1 Programación no Estructurada.

Este tipo de programación se basa en una secuencia de código capaz de modificar los datos globales en el transcurso del programa, código que no tendrá una estructura determinada. Donde existen saltos que irían a cualquier lugar del código, convirtiéndose este en un laberinto indescifrable para muchos, sin saber cuándo termina la ejecución del mismo. [9]

1.3.4.2 **Programación Procedimental**.

Con la programación procedimental se pueden combinar las secuencias de instrucciones repetibles en un solo lugar. Después de que la secuencia es procesada, el flujo de control procede exactamente después de la posición donde la llamada fue hecha.

Al introducir parámetros así como procedimientos de procedimientos (subprocedimientos) los programas pueden ser escritos en forma más estructurada y libres de errores. Por ejemplo, si un procedimiento ya es correcto, cada vez que es usado produce resultados correctos.

Por consecuencia, en caso de errores, se puede reducir la búsqueda a aquellos lugares que todavía no han sido revisados. De este modo, un programa puede ser visto como una secuencia de llamadas a procedimientos.

El programa principal es responsable de pasar los datos a las llamadas individuales, los datos son procesados por los procedimientos y, una vez que el programa ha terminado, los datos resultantes son presentados.[9]

1.3.4.3 Programación Modular

Uno de los métodos más conocidos para resolver un problema es dividirlo en problemas más pequeños, llamados subproblemas, de esta manera se obtiene una solución de forma más sencilla. Esta técnica se utiliza mucho en programación, y se le suele llamar diseño descendente, metodología del divide y vencerás o programación top-down.

Es evidente que si esta metodología lleva a tratar con subproblemas, entonces también se tenga la necesidad de poder crear y trabajar con subprogramas para resolverlos. A estos subprogramas se les suele llamar módulos, de ahí viene el nombre de *programación modular*.[10]

1.3.4.4 La programación orientada a objetos (POO)

La programación orientada a objetos, intenta simular el mundo real a través del significado de objetos que contiene características y funciones. Como su mismo nombre indica, la programación orientada a objetos se basa en la idea de un objeto, que es una combinación de variables locales y procedimientos llamados métodos que juntos conforman una entidad de programación.

A continuación se muestra una breve reseña de la aplicación de algunos conceptos en este tipo de programación.

- *encapsular*: significa, reunir y controlar el grupo resultante como un todo y no individualmente.
- *la abstracción*: es un término externo al objeto, que controla la forma en que es visto por los demás.
- *la modularidad:* se considera de la siguiente manera: un programa grande siempre será más complicado que la suma de varios programas pequeños, con lo que se considera ventajoso dividir un gran sistema en diversos módulos.
- *la jerarquía*: consiste en la clasificación y organización de las abstracciones según su naturaleza. El más claro ejemplo de jerarquía es la herencia.
- *la herencia:* se define como una jerarquía de extracciones, y la relación entre clases, donde se comparte la estructura y el comportamiento de una o más clase considerada como clases superiores o una superclase, con lo cual se resume que la herencia es una unidad independiente por si misma heredada de una abstracción o superclase.[11]

1.3.5 Lenguajes y plataformas de desarrollo

En computación, un programa es una secuencia de instrucciones que permiten a un ordenador procesar una información conocida como datos de entrada (input) para producir una información de salida (output) o resultados. Esas instrucciones pertenecen a (o están escritas en) un lenguaje de programación determinado.

Un lenguaje de programación es una construcción mental del ser humano para expresar programas. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos monosémicos (es decir, con sentido único) y una regla principal que resume las demás. Para que ésta construcción mental sea operable en un computador debe existir otro programa que controle la validez o no de lo escrito. A éste se le llama traductor.

Para el desarrollo de aplicaciones web existen dos grandes grupos lenguajes de programación: Lenguajes del lado del cliente y lenguajes del lado del servidor.

1.3.5.1 Lenguaje del lado del cliente:

Dentro del grupo de lenguajes del lado del cliente algunos de los más utilizados a nivel mundial son: JavaScript, XSLT y el Visual Basic Script, estos dos últimos al combinarse con el HTML forman lo que se conoce como DHTML, salida estándar dinámica o HTML dinámico. Para crear páginas interactivas se utiliza la metodológica AJAX (XML y JavaScript asíncronos).

1.3.5.1.1 JavaScript

Es un lenguaje de programación del lado del cliente, pues en el navegador es donde se produce la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. Es utilizado por millones de páginas webs para validar formularios, crear cookies, detectar navegadores y mejorar el diseño, su fácil aprendizaje lo hace un lenguaje muy demandado.

1.3.5.1.2 Visual Basic Script

Es un lenguaje que, a diferencia de JavaScript, es solamente compatible con Internet Explorer, aunque posee toda la funcionalidad que brinda JavaScript.

1.3.6 Lenguaje del lado del servidor.

Dentro del grupo de lenguajes del lado del servidor los más usados a nivel mundial son: C#, PHP, Java, JSP, PERL, etc. A través de ellos los desarrolladores implementan la lógica de negocio dentro del servidor, además de los accesos a los distintos Sistemas de Gestión de Bases de Datos (SGBD).

1.3.6.1 PHP

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas, similar al ASP de Microsoft o el JSP de Sun, embebido en páginas HTML y ejecutado en el servidor..

La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. No es un lenguaje de marcas como podría ser HTML, XML o WML. Está más cercano a JavaScript o a C, para aquellos que conocen estos lenguajes.[12]

1.3.6.2 Java

Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. También posee otras características muy importantes: [13]

- Es un lenguaje que es compilado, generando ficheros de clases compilados, pero estas clases compiladas, son en realidad interpretadas por la máquina virtual de java. Siendo la máquina virtual de java la que mantiene el control sobre las clases que se estén ejecutando.
- Es un lenguaje multiplataforma: El mismo código java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual java.
- Es un lenguaje seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.

- Gracias al API de Java se amplia el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas, crear aplicaciones visuales al estilo Windows.

1.3.6.3 Visual C# 2005

Microsoft Visual C# 2005 es un lenguaje de programación diseñado para crear una amplia gama de aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Con sus diversas innovaciones, C# permite desarrollar aplicaciones rápidamente y mantiene la expresividad y elegancia de los lenguajes de tipo C.[14]

1.3.7 Tecnologías de desarrollo

1.3.7.1 ASP.NET

ASP .NET significa Páginas Activas de Servidor .NET (*Active Server Pages .NET*). Es una tecnología para generar páginas dinámicas en el servidor y enviarlas al cliente (navegador Web) que las ha solicitado, ejecutando previamente el código que contienen (código Visual Basic, C#, etc.) y convirtiendo el resultado a código HTML, que es el único que puede interpretar adecuadamente el cliente.

A diferencia de sus predecesores, 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 la 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.[15]

1.3.7.2 Ajax

Ajax no es un lenguaje, exactamente su nombre viene dado por el acrónimo de Asynchronous JavaScript And XML, es realmente muchas tecnologías, cada una florecida por su propio mérito, uniéndose en poderosas nuevas formas. AJAX incorpora:

- Presentación basada en estándares usando XHTML y CSS.
- Exhibición e interacción dinámicas usando el Document Object Model.

- Intercambio y manipulación de datos usando XML y XSL.
- Recuperación de datos asincrónica usando XMLHttpRequest.
- JavaScript para manipular estas tecnologías.[16]

1.3.7.3 Lenguaje de Marcas Extensible (XML)

XML es un lenguaje que ofrece un formato para la descripción de datos estructurados lo que facilita declaraciones de contenido más precisas y resultados de búsquedas más significativos en varias plataformas. Además, XML habilitará una nueva generación de aplicaciones para ver y manipular datos basadas en el Web.

XML ofrece una representación estructural de los datos que se puede implementar ampliamente y es fácil de distribuir. Garantiza que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes. La interoperabilidad resultante está creando rápidamente una nueva generación de aplicaciones de comercio electrónico en la Web.

XML proporciona un estándar de datos que puede codificar el contenido, la semántica y los esquemas de una gran variedad de casos, desde los más simples a los más complejos, sirve para marcar lo siguiente:

- Un documento normal.
- Un registro estructurado, como un registro de citas o un pedido de compra.
- Un objeto con datos y métodos, como el formulario permanente de un objeto Java o de un control ActiveX.
- Un registro de datos, como el conjunto de resultados de una consulta.
- Metacontenido sobre un sitio Web, como el formato de definición de canal (CDF).
- Representaciones gráficas, como la interfaz de usuario de una aplicación.
- Entidades y tipos de esquema estándar.
- Todos los vínculos entre datos y personas que hay en el Web. [17]

1.3.7.4 Lenguaje de Estilo Extensible (XSL)

XSL es una Tecnología XML de hojas de estilos que sirve para mostrar documentos XML, darles formato de presentación. La tecnología XSL sirve para transformar documentos XML en otros XML. Éste, permite la manipulación de la información XML.

1.3.8 Plataformas

1.3.8.1 Net Framework v2.0

Microsoft .NET Framework versión 2.0 Redistributable Package instala el entorno en tiempo de ejecución y los archivos asociados de .NET Framework necesarios para ejecutar aplicaciones desarrolladas para .NET Framework v2.0.

.NET Framework versión 2.0 mejora la escalabilidad y el rendimiento de aplicaciones gracias a características mejoradas como el almacenamiento en caché, el desarrollo de aplicaciones y la actualización con ClickOnce; además, es compatible con la gama más amplia de exploradores y dispositivos con servicios y controles ASP.NET 2.0.[18]

El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.
- Ofrecer un entorno de ejecución de código que fomente la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos.
- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.
- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

El Common Language Runtime o CLR es el motor de la plataforma. NET, encargado de gestionar la ejecución de las aplicaciones .NET, a las cuales ofrece numerosos servicios para simplificar su desarrollo, favoreciendo con ello su fiabilidad y seguridad. Las principales características y servicios que proporciona son los siguientes: [19]

- Modelo de programación consistente.
- Modelo de programación sencillo.
- Ejecución multiplataforma.
- Aislamiento de procesos.
- Tratamiento de excepciones.
- Soporte multihilo.
- Distribución transparente.
- Seguridad avanzada.
- Gestión de memoria.
- Interoperabilidad con código antiguo.

La biblioteca de clases de .NET Framework es una colección de tipos reutilizables que se integran estrechamente con Common Language Runtime. La biblioteca de clases está orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Esto ocasiona que los tipos de .NET Framework sean sencillos de utilizar y reduce el tiempo asociado con el aprendizaje de las nuevas características de .NET Framework. Además, los componentes de terceros se pueden integrar sin dificultades con las clases de .NET Framework.

Permite realizar diversas tareas de programación comunes, como son la administración de cadenas, recopilación de datos, conectividad de bases de datos y acceso a archivos. Además de estas tareas habituales, la biblioteca de clases incluye tipos adecuados para diversos escenarios de desarrollo especializados. Por ejemplo, puede utilizar .NET Framework para desarrollar los siguientes tipos de aplicaciones y servicios: [18]

- Aplicaciones de consola.
- Aplicaciones GUI de Windows (Windows Forms).
- Aplicaciones de ASP.NET.

- Servicios Web XML.
- Servicios de Windows.

1.3.8.2 Plataforma Mono

Es una plataforma de desarrollo de código abierto basada en .NET Framework para ejecutar y desarrollar aplicaciones modernas basadas en los estándares ECMA/ISO. Puede ejecutar aplicaciones hechas para los Framework .NET y Java.

Esta plataforma ofrece: [20]

- Un entorno avanzado de desarrollo para escribir aplicaciones Linux con productividad sin precedentes.
- APIs exhaustivos para la entrega de completas aplicaciones para cliente, aplicaciones para servicios Web y para servidores.
- Capacidades de despliegue de plataforma cruzada, dando soporte a Linux, Microsoft Windows NT/XP y otros sistemas UNIX de diversas arquitecturas.
- Librería de programación de plataforma cruzada Gtk# GUI, que permite a los desarrolladores acceder a Linux, Windows y el MacOS X con un simple código base de cualquiera de los lenguajes de programación compatibles de Mono.
- Soporte para varios lenguajes tales como VisualBasic, Python, JScript y Java. Mono incluye soporte Java, mediante el proyecto de código abierto iKVM, haciendo de este popular lenguaje un vehículo capaz de apalancar la tecnología Mono.

1.3.9 Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE) consiste básicamente en un *software* que previamente ha sido instalado en la máquina del cliente y cuyo principal objetivo es el desarrollo de otro *software*.

Visual Studio proporciona las herramientas que necesita para diseñar, desarrollar, depurar e implementar aplicaciones Web, Servicios Web XML y aplicaciones cliente tradicionales, comparte un entorno de desarrollo integrado (IDE) único.

El IDE está compuesto por varios elementos: la barra de herramientas Menú, la barra de herramientas Estándar, varias ventanas de herramientas acopladas o que se ocultan automáticamente en la parte izquierda, derecha e inferior, así como el espacio del editor. Las ventanas de herramientas, menús y barras de herramientas disponibles en un momento dado dependen del tipo de proyecto o archivo en el que esté trabajando.[21]

1.3.9.1 SharpDevelop

SharpDevelop es un entorno de desarrollo integrado para la plataforma .NET. Soporta las versiones del FrameWork de Microsoft y de Ximian. Soporta desarrollo de interfaces, clases, namespaces y proyectos en C# y VB.NET, además de permitir importar los proyectos creados con Microsoft Visual Studio .NET.[22]

SharpDevelop incorpora:

- Diseñador de Formularios.
- Completado de Código.
- Auto-insertado de Código.
- Conversor de Código C# a VB.Net y viceversa.
- Importar/Exportar Soluciones VS.NET a Visual Studio .NET.
- Plegado de Código ("Folding").
- Visor gráfico para realizar pruebas con NUnit.
- Analizador del Código Ensamblador.
- Vista previa de Documentación en XML.
- También incluye sintaxis coloreada, paréntesis inteligentes, bookmarks, plantillas, herramientas para expresiones regulares, asistentes, exportación HTML, visor de clases, integración con NDoc, integración con Nprof, etc.

1.3.9.2 Visual Studio 2005

Visual Studio presenta un nuevo diseñador de páginas Web que incluye muchas mejoras para la creación y edición de páginas Web de ASP.NET y páginas HTML, además, varias funciones de productividad como son la Configuración del IDE, importar y exportar configuraciones, listas de tareas, lista de errores, teclas de método abreviado Brief y Emacs.

. Proporciona una forma fácil y rápida de crear páginas de formularios Web Forms.

La vista Diseño del diseñador HTML facilita el diseño WYSIWYG de páginas Web. La edición basada en tareas mediante etiquetas inteligentes le guía durante la ejecución de los procedimientos más comunes con controles, como el enlace de datos y la asignación de formato.

Puede editar visualmente las nuevas páginas principales de ASP.NET. La edición de plantillas se ha mejorado para facilitar el trabajo con controles de datos, así como con nuevos controles como el control Login. Editar tablas HTML para el diseño o mostrar la información en columnas ahora es más fácil e intuitivo. [21]

En el presente capítulo se han analizado un conjunto de aplicaciones estrechamente relacionadas con el campo de acción. Después del análisis se ha concluido que las aplicaciones internacionales no se corresponden con el sistema de salud cubano, además de las desventajas, de no permitir adaptaciones por ser propietarias y el despliegue de las mismas constituye una gran inversión de recursos. La aplicación existente en la actualidad en nuestro país no responde a las necesidades de las instituciones oftalmológicas, pues carece de varios servicios.

Se ha analizado también un grupo de tecnologías y lenguajes candidatos para el desarrollo de la propia aplicación, arribando a la conclusión en conjunto con el arquitecto del sistema el uso de: la plataforma .NET, ASP.NET para la implementación de las interfaces web, C# como lenguaje general para la lógica de negocio y lógica de acceso a datos, como lenguaje de desarrollo del lado del cliente se usará JavaScript y AJAX. Usando como IDE de desarrollo Visual Studio 2005. Compilado la aplicación sobre el framework .NET, para poder usar las bibliotecas de clases y el CLR de .NET. Aprovechando de cada una la factibilidad y ventajas expuestas, siendo estas de significativa importancia para el desarrollo de la aplicación.

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

En este capítulo se aborda el análisis del diseño propuesto por los analistas. Además se analizan las posibles implementaciones de determinados módulos y componentes así como la reutilización de los anteriores. Se realizarán descripciones de clases, los tipos de datos y las soluciones que se implementan para dar respuesta al problema.

2.1 Valoración crítica del diseño propuesto por el analista.

Después de analizado el diseño propuesto por los analistas, se logró definir las principales clases para el correcto funcionamiento del mismo, así como los atributos y los métodos que deben tener las mismas, reflejando una idea clara de lo que se quiere implementar. Además de 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 obtuvo la información necesaria para conocer el orden de las acciones a implementar.

El diseño propuesto se creó basado en el seguimiento de patrones, los cuales se traducen en soluciones factibles a los problemas específicos del diseño orientado a objetos, concibiendo así una implementación más clara del sistema bajo determinados patrones de los grupos 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.[23]

Es válido detallar que la implementación de este sistema no cumple con el patrón de Bajo Acoplamiento, por peticiones del cliente, se consume el servicio web SAAA(EMP) brindado por los compañeros de la empresa SOFTEL, básicamente el sistema basa su seguridad en dicho componente, por lo cual constituye un pilar en su funcionamiento; aunque cuando se controle el sistema y el ya referenciado servicio en la misma instalación hospitalaria por un único administrador se minimizan drásticamente las posibles incongruencias.

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 Factory method y Abstract Factory, facilitando este último 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.[23]

2.2 Análisis de posibles implementaciones, componentes o módulos ya existentes que puedan ser rehusados.

La Capa Intermedia (Middleware) forma parte de la solución del problema, a través de esta subcapa se accede al repositorio de datos, está encargada de ejecutar la lógica de acceso a datos, contiene dos paquetes de clase:

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.

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, 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).

Para enlazar el Middleware con el Acceso a Datos que está en el servidor de BD se requiere el uso la *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 está conformado por un conjunto de funciones programadas en lenguaje NpgSql.[23]

El Sistema de Información Hospitalaria está compuesto por varios módulos, de los que el Bloque Quirúrgico Oftalmológico necesita intercambiar información.

Entre estos módulos están:

Inscripción-Admisión: A través de este módulo se realizan las búsquedas de los pacientes. Una vez seleccionado el paciente al cual se le realizará la consulta, muestra toda la información correspondiente a este, como por ejemplo: número de identificación o carnet de identidad, nombre y apellidos, sexo, edad, municipio, provincia, país, sala y cama en caso de estar ingresado, entre otros.

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

Farmacia: Abastecedor y controlador de los medicamentos que están disponibles, así como los materiales gastables implicados en cada una de las intervenciones quirúrgicas.

2.3 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. La aplicación utilizará el Servicio Web SAAA(EMP), donde la información trasmitida estará regida por el formato Web Services Description Language(WSDL), un formato XML que se utiliza para describir servicios Web.

La base de datos es accedida de forma directa mediante clases controladoras y los componentes rehusados son integrados mediante interfaces sencillas.[23]

El objeto ObjectDataSource es un control de origen de datos de ASP.NET que representa un objeto de nivel medio orientado a datos o un objeto de interfaz de datos con controles enlazados a datos. Se puede utilizar el control ObjectDataSource junto con un control enlazado a datos para presentar, editar y ordenar datos de una página Web que contenga poco código o no lo contenga.

Este componente es utilizado para el módulo de configuración, afianzado en ventajas como:

- Permite a los desarrolladores utilizar un control de origen de datos de ASP.NET y a la vez retener su arquitectura de aplicación de tres niveles.

- Utiliza la reflexión para crear instancias de objetos comerciales y para llamar a los métodos que contienen para recuperar, actualizar, insertar y eliminar datos.[24]

De esta forma disminuye el número de líneas de código y a su vez el nivel de errores para el módulo en cuestión.

La aplicación además garantizándole una mayor confiabilidad al cliente, será integrada con un módulo de administración, el cual tramitará toda la seguridad del sistema, basando su proceso de integración en el consumo del ya mencionado componente SAAA (EMP).

2.4 Descripción de los algoritmos no triviales a implementar.

Los exámenes especializados es uno de los servicios que será automatizado por el Bloque Quirúrgico Oftalmológico. Los mismos pueden ser accedidos desde las diferentes consultas o desde el menú principal de la aplicación, siempre que se seleccione un paciente inicialmente.

Una vez seleccionado el paciente, el usuario seleccionará el tipo de examen a gestionar, entre ellos se encuentra Refracción (VAP), en el mismo, el especialista, en el momento de guardar los datos, la aplicación chequeará el valor del cilindro, de ser positivo, la misma lanzará un mensaje de alerta al usuario, quien podrá valorar si desea insertar un valor positivo para el cilindro o no, de decidir un valor positivo, la aplicación realizará una serie de transformaciones en los datos recibidos.

2.4.1 Análisis de complejidad.

La Complejidad Ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Esta métrica se basa en la representación gráfica del flujo de control del programa. De dicho análisis se desprende una medida cuantitativa de la dificultad de prueba y una indicación de la fiabilidad final.

Cuando se utiliza en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y proporcionando el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. Es una de las métricas de software más ampliamente aceptada, ya que ha sido concebida para ser independiente del lenguaje.

Se ha medido un gran número de programas, de modo de establecer rangos de complejidad que ayuden al ingeniero de software a determinar la estabilidad y riesgo inherente de un programa. La

medida resultante puede ser utilizada en el desarrollo, mantenimiento y reingeniería para estimar el riesgo, costo y estabilidad.

Algunos estudios experimentales indican la existencia de distintas relaciones entre la métrica de McCabe y el número de errores existentes en el código fuente, así como el tiempo requerido para encontrar y corregir esos errores. Se suele comparar la complejidad ciclomática obtenida contra un conjunto de valores límite como se observa en la **tabla 1**. [25]

Complejidad Ciclomática	**Evaluación del Riesgo**
1-10	Programa Simple, sin mucho riesgo
11-20	Más complejo, riesgo moderado
21-50	Complejo, Programa de alto riesgo
50	Programa no testeable, Muy alto riesgo

Tabla 1: Complejidad ciclomática vs evaluación de riesgo.

La complejidad ciclomática puede ser aplicada en varias áreas incluyendo:

- *Análisis de Riesgo en desarrollo de código*:

Mientras el código esta en desarrollo, su complejidad puede ser medida para estimar el riesgo inherente.

- *Análisis de riesgo de cambio durante la fase de mantenimiento*:

La complejidad del código tiende a incrementarse a medida que es mantenido durante el tiempo. Midiendo la complejidad antes y después de un cambio propuesto, puede ayudar a decidir cómo minimizar el riesgo del cambio.

- *Planificación de Pruebas*:

El análisis matemático ha demostrado que la complejidad ciclomática indica el número exacto de casos de prueba necesarios para probar cada punto de decisión en un programa.

- *Reingeniería*:

Provee conocimiento de la estructura del código operacional de un sistema. El riesgo involucrado en la reingeniería de una pieza de código está relacionado con su complejidad.

En el análisis de riesgo de desarrollo para conocer la complejidad del algoritmo es necesario calcular la complejidad ciclomática del mismo, para hacer dicho cálculo es necesario primero tener el código o el diseño del algoritmo, luego enmarcar cada instrucción del código con un número, que representa cada lugar del camino que puede seguir la secuencia del algoritmo. A continuación se representa el código con sus instrucciones enmarcadas:[25]

```
protected void ConvertirCilindroPos()
  { 1
     double aux = 0; 1
     double aux1 = 0; 1
     bool ya = false; 1
     aux = double.Parse(TextBoxEsferaOD.Text) +  double.Parse(TextBoxCilindroOD.Text); 1
     TextBoxEsferaOD.Text = aux.ToString(); 1
     TextBoxCilindroOD.Text = "-" + TextBoxCilindroOD.Text; 1
     if (double.Parse(TextBoxEjeOD.Text) <= 90) 2
     {
        aux1 = double.Parse(TextBoxEjeOD.Text) + 90; 3
        TextBoxEjeOD.Text = aux1.ToString(); 3
        ya = true; 3

     }
     if (double.Parse(TextBoxEjeOD.Text) > 90 && ya==false) 4
     {
        aux1 = double.Parse(TextBoxEjeOD.Text) - 90; 5
        TextBoxEjeOD.Text = aux1.ToString(); 5
        ya = false; 5
     }

  } 6
```

Para poder dar una definición acabada de la complejidad ciclomática, es necesario primero introducir una sencilla notación para la representación del flujo de control, denominada Grafos de Flujo de Control de un programa.

Cada círculo se denomina NODO y representa una o más sentencias procedimentales. Las flechas se llaman ARISTAS y representan flujo de control. Una arista debe terminar en un nodo, aún cuando éste no represente ninguna sentencia procedimental. Las áreas delimitadas por aristas y nodos se denominan regiones. Cuando se contabilizan las regiones se debe incluir el área exterior del grafo como una región más.

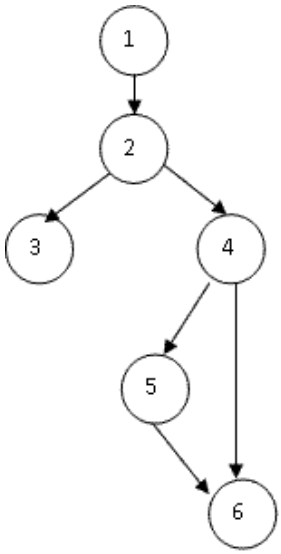

Imagen 1. Grafo de flujo de la función ConvertirCilindroPos.

Una vez obtenido el grafo de flujo de la función en cuestión se puede decir que existen varias formas de calcular la complejidad ciclomática de un programa a partir del Grafo de flujo representado anteriormente.

- La complejidad ciclomática, V(G), de un grafo de flujo G se define como:

V(G) = Aristas – Nodos + 2

V(G) = 7-6+2 = 3

- La complejidad ciclomática, V(G), de un grafo de flujo G se define como:

V(G) = Nodos Predicado + 1

V(G)= 2 + 1 = 3

- El número de regiones del grafo coincide con la complejidad ciclomática, V(G).

VG = Número de regiones

VG = 3

Esta complejidad ciclomática determina el número de casos de prueba que deben ejecutarse para garantizar que todas las sentencias de un programa se han ejecutado al menos una vez, y que cada condición se habrá ejecutado en sus vertientes verdadera y falsa.

Realizado el cálculo por las tres vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática de 3, la evaluación de riesgo demuestra que es un programa simple, sin mucho riesgo.

2.5 Estándares de codificación

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. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente.

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. El mantenimiento 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 el mantenimiento 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.

Aunque el propósito principal para llevar a cabo revisiones del código a lo largo de todo el desarrollo es localizar defectos en el mismo, las revisiones también pueden afianzar los estándares de codificación de manera uniforme. La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final del proyecto de software. No es práctico, ni prudente, imponer un estándar de codificación una vez iniciado el trabajo.

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.[26]

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

El uso de los estándares de codificación en la investigación, están presentes de la siguiente manera.

2.5.1-Notación Camello.

Se usa para denotar variables y parámetros. En esta notación, si el identificador es una palabra simple se escribe todo con minúscula, pero si es compuesta, la primera letra de todas las palabras que viene a continuación de la primera comienza con mayúscula.

Ejemplo:

private Int32 *idPersona*;

private Int32 *numeroConsulta*;

public List<Consulta> BuscarConsulta(Int32 *numeroConsulta,* DateTime *fechaConsulta*)

```
{
        //…
}
```

2.5.2-Notación Pascal

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

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 es pequeño (tres o menos caracteres) se escribirá enteramente con mayúscula. Ejemplo:

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

2.5.2.1 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
{
        //…
}
```

2.5.2.2 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 List<DatosPersona> BuscarPaciente (DatosPersona persona)
{
    //…
}
```

2.5.2.3 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
{
    get
    {
        return datosConsulta;
    }
    set
    {
        datosConsulta = value;
    }
}
```

2.5.2.4 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

2.5.2.5 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[14].Activa )
        {
                //….
        }
}
```

Como se pudo observar 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.[23]

2.6 Descripción de las nuevas clases u operaciones necesarias.

2.6.1 Entidades del negocio

Estas entidades fueron creadas siguiendo el patrón de mapeo de datos, que propone crear un objeto por cada entidad persistente (tabla de la BD).

Nombre: Refracción	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenesOftalmologicos	Int32
agudezaVisualSinCorreccion	Double
esfera	Double
cilindro	Double
eje	Double
agudezaVisualConCorreccion	Double
add	Double
dp	Double
Para cada responsabilidad:	
Nombre:	Refraccion
Descripción:	Constructor de la clase.
Nombre:	IdExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el idExamenesOftalmologicos
Nombre:	AgudezaVisualSinCorreccion
Descripción:	Propiedad para mostrar y modificar la agudezaVisualSinCorreccion

Nombre:	Esfera
Descripción:	Propiedad para mostrar y modificar la esfera
Nombre:	Cilindro
Descripción:	Propiedad para mostrar y modificar el cilindro
Nombre:	Eje
Descripción:	Propiedad para mostrar y modificar el eje
Nombre:	AgudezaVisualConCorreccion
Descripción:	Propiedad para mostrar y modificar la agudezaVisualConCorreccion
Nombre:	Add
Descripción:	Propiedad para mostrar y modificar la add
Nombre:	Dp
Descripción:	Propiedad para mostrar y modificar la dp

Tabla 1: Clase entidad "Refraccion"

Nombre: Queratometria	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenesOftalmologicos	Int32
numeroConsulta	Int32
meridianoDebil	Double
meridianoFuerete	Double
ejeFuerte	Double

ejeDebil	Double
Para cada responsabilidad:	
Nombre:	Queratometria
Descripción:	Constructor de la clase.
Nombre:	IdExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el idExamenesOftalmologicos
Nombre:	NumeroConsulta
Descripción:	Propiedad para mostrar y modificar el numeroConsulta
Nombre:	MeridianoDebil
Descripción:	Propiedad para mostrar y modificar el meridianoDebil
Nombre:	MeridianoFuerete
Descripción:	Propiedad para mostrar y modificar el meridianoFuerte
Nombre:	EjeFuerte
Descripción:	Propiedad para mostrar y modificar el ejeFuerte
Nombre:	EjeDebil
Descripción:	Propiedad para mostrar y modificar el ejeDebil

Tabla 2: Clase entidad "Queratometria"

Nombre: Biometria	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenesOftalmologicos	Int32
numeroConsulta	Int32

profundidadCamaraAnterior	Double
grosarioDelCristalino	Double
longitudAccial	Double
Para cada responsabilidad:	
Nombre:	Biometria
Descripción:	Constructor de la clase.
Nombre:	IdExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el idExamenesOftalmologicos
Nombre:	numeroConsulta
Descripción:	Propiedad para mostrar y modificar el numeroConsulta
Nombre:	ProfundidadCamaraAnterior
Descripción:	Propiedad para mostrar y modificar la profundidadCamaraAnterior
Nombre:	GrosarioDelCristalino
Descripción:	Propiedad para mostrar y modificar el grosarioDelCristalino
Nombre:	LongitudAccial
Descripción:	Propiedad para mostrar y modificar la longitudAccial

Tabla 3: Clase entidad "Biometria"

Nombre: TensionOcular	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenesOftalmologicos	Int32
tonometroNoContacto	Int32

aplanacionGoldman	Double
aplanacionPerkins	Double
pio	Double
tonometriaContornoDinamico	Double
digitalNormal	Boolean
digitalHipertenso	Boolean

Para cada responsabilidad:	
Nombre:	TensionOcular
Descripción:	Constructor de la clase.
Nombre:	IdExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el idExamenesOftalmologicos
Nombre:	TonometroNoContacto
Descripción:	Propiedad para mostrar y modificar el tonometroNoContacto
Nombre:	AplanacionGoldman
Descripción:	Propiedad para mostrar y modificar la aplanacionGoldman
Nombre:	AplanacionPerkins
Descripción:	Propiedad para mostrar y modificar la aplanacionPerkins
Nombre:	Pio
Descripción:	Propiedad para mostrar y modificar el pio
Nombre:	TonometriaContornoDinamico
Descripción:	Propiedad para mostrar y modificar la tonometriaContornoDinamico
Nombre:	DigitalNormal
Descripción:	Propiedad para mostrar y modificar la digitalNormal

Nombre:	DigitalHipertenso
Descripción:	Propiedad para mostrar y modificar la digitalHipertenso

Tabla 4: Clase entidad "TensionOcular"

Nombre: ExamenRefraccionDinamica	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenesOftalmologicos	Int32
agudezaVisualSinCorreccion	Double
esfera	Double
cilindro	Double
eje	Double
agudezaVisualConCorreccion	Double
add	Double
dp	Double
optotipologmar	Boolean
snellen	Boolean
pruebaFinal	Boolean

Para cada responsabilidad:	
Nombre:	Refraccion
Descripción:	Constructor de la clase.
Nombre:	IdExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el idExamenesOftalmologicos

Nombre:	AgudezaVisualSinCorreccion
Descripción:	Propiedad para mostrar y modificar la agudezaVisualSinCorreccion
Nombre:	Esfera
Descripción:	Propiedad para mostrar y modificar la esfera
Nombre:	Cilindro
Descripción:	Propiedad para mostrar y modificar el cilindro
Nombre:	Eje
Descripción:	Propiedad para mostrar y modificar el eje
Nombre:	AgudezaVisualConCorreccion
Descripción:	Propiedad para mostrar y modificar la agudezaVisualConCorreccion
Nombre:	Add
Descripción:	Propiedad para mostrar y modificar la add
Nombre:	Dp
Descripción:	Propiedad para mostrar y modificar la dp
Nombre:	Optotipologmar
Descripción:	Propiedad para mostrar y modificar el optotipologmar
Nombre:	Snellen
Descripción:	Propiedad para mostrar y modificar la snellen
Nombre:	PruebaFinal
Descripción:	Propiedad para mostrar y modificar la pruebaFinal

Tabla 5: Clase entidad “ExamenRefraccionDinamica”

Nombre: ExamenRefraccionIndicada	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenesOftalmologicos	Int32
agudezaVisualConCristales	Double
esfera	Double
cilindro	Double
eje	Double
agudezaVisualConCorreccion	Double
add	Double
dp	Double
Para cada responsabilidad:	
Nombre:	Refraccion
Descripción:	Constructor de la clase.
Nombre:	IdExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el idExamenesOftalmologicos
Nombre:	AgudezaVisualConCristales
Descripción:	Propiedad para mostrar y modificar la AgudezaVisualConCristales
Nombre:	Esfera
Descripción:	Propiedad para mostrar y modificar la esfera
Nombre:	Cilindro
Descripción:	Propiedad para mostrar y modificar el cilindro
Nombre:	Eje

Descripción:	Propiedad para mostrar y modificar el eje
Nombre:	AgudezaVisualConCorreccion
Descripción:	Propiedad para mostrar y modificar la agudezaVisualConCorreccion
Nombre:	Add
Descripción:	Propiedad para mostrar y modificar la add
Nombre:	Dp
Descripción:	Propiedad para mostrar y modificar la dp

Tabla 6: Clase entidad "ExamenRefraccionIndicada"

Nombre: ExamenLensometro	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenesOftalmologicos	Int32
esfera	Double
cilindro	Double
eje	Double
add	Double
dp	Double
Para cada responsabilidad:	
Nombre:	Refraccion
Descripción:	Constructor de la clase.
Nombre:	IdExamenesOftalmologicos

Descripción:	Propiedad para mostrar y modificar el idExamenesOftalmologicos
Nombre:	Esfera
Descripción:	Propiedad para mostrar y modificar la esfera
Nombre:	Cilindro
Descripción:	Propiedad para mostrar y modificar el cilindro
Nombre:	Eje
Descripción:	Propiedad para mostrar y modificar el eje
Nombre:	Add
Descripción:	Propiedad para mostrar y modificar la add
Nombre:	Dp
Descripción:	Propiedad para mostrar y modificar la dp

Tabla 7: Clase entidad "ExamenLensometro"

Nombre: PaquimetriaCorneal	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenesOftalmologicos	Int32
grosorCornea	Int32
Para cada responsabilidad:	
Nombre:	PaquimetriaCorneal
Descripción:	Constructor de la clase.
Nombre:	IdExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el idExamenesOftalmologicos

Nombre:	GrosorCornea
Descripción:	Propiedad para mostrar y modificar el grosorCornea

Tabla 8: Clase entidad "PaquimetriaCorneal"

Nombre: ExamenesOftalmologicos	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenesOftalmologicos	Int32
idPersona	Int32
idOjos	Int32
iolMaster	Boolean
fechaHoraExamen	DateTime
Para cada responsabilidad:	
Nombre:	ExamenesOftalmologicos
Descripción:	Constructor de la clase.
Nombre:	IdExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el idExamenesOftalmologicos
Nombre:	IdPersona
Descripción:	Propiedad para mostrar y modificar el idPersona
Nombre:	IdOjos
Descripción:	Propiedad para mostrar y modificar el idOjos
Nombre:	IolMaster
Descripción:	Propiedad para mostrar y modificar el iolMaster

Nombre:	FechaHoraExamen
Descripción:	Propiedad para mostrar y modificar la fechaHoraExamen

Tabla 9: Clase entidad "ExamenesOftalmologicos"

Nombre: AntecedentesPatologicosOftalmologicos	
Tipo de clase: Entidad	
Atributo	**Tipo**
idAntecedentesOftalmologicos	Int32
numeroConsulta	Int32
idPersona	Int32
Para cada responsabilidad:	
Nombre:	AntecedentesPatologicosOftalmologicos
Descripción:	Constructor de la clase.
Nombre:	IdAntecedentesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el idAntecedentesOftalmologicos
Nombre:	NumeroConsulta
Descripción:	Propiedad para mostrar y modificar el numeroConsulta
Nombre:	IdPersona
Descripción:	Propiedad para mostrar y modificar el idPersona

Tabla 10: Clase entidad "AntecedentesPatologicosOftalmologicos"

Nombre: ExamenCristalino	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamen	Int32
idExamenOftalmologicoAnatomia	Int32
idTipoCatarata	Int32
idTipoLente	Int32
Para cada responsabilidad:	
Nombre:	ExamenCristalino
Descripción:	Constructor de la clase.
Nombre:	IdExamen
Descripción:	Propiedad para mostrar y modificar el idExamen
Nombre:	IdExamenOftalmologicoAnatomia
Descripción:	Propiedad para mostrar y modificar el idExamenOftalmologicoAnatomia
Nombre:	IdTipoCatarata
Descripción:	Propiedad para mostrar y modificar el idTipoCatarata
Nombre:	IdTipoLente
Descripción:	Propiedad para mostrar y modificar el idTipoLente

Tabla 11: Clase entidad “ExamenCristalino”

Nombre: ExamenesAnexos	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenOftalmologicoAnatomia	Int32
idParámetroAnatomia	Int32
Para cada responsabilidad:	
Nombre:	ExamenesAnexos
Descripción:	Constructor de la clase.
Nombre:	IdExamenOftalmologicoAnatomia
Descripción:	Propiedad para mostrar y modificar el idExamenOftalmologicoAnatomia
Nombre:	IdParámetroAnatomia
Descripción:	Propiedad para mostrar y modificar el idParámetroAnatomia

Tabla 12: Clase entidad "ExamenesAnexos"

Nombre: ExamenFondoOjo	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenOftalmologicoAnatomia	Int32
idParámetroFondoOjo	Int32
Para cada responsabilidad:	
Nombre:	ExamenFondoOjo
Descripción:	Constructor de la clase.
Nombre:	IdExamenOftalmologicoAnatomia

Descripción:	Propiedad para mostrar y modificar el idExamenOftalmologicoAnatomia
Nombre:	IdParámetroFondoOjo
Descripción:	Propiedad para mostrar y modificar el idParámetroFondoOjo

Tabla 13: Clase entidad "ExamenFondoOjo"

Nombre: ExamenMotilidadOcular	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenMotilidad	Int32
idExamenOftalmologicoAnatomia	Int32
idOtropia	Int32
Para cada responsabilidad:	
Nombre:	ExamenMotilidadOcular
Descripción:	Constructor de la clase.
Nombre:	IdExamenMotilidad
Descripción:	Propiedad para mostrar y modificar el idExamenMotilidad
Nombre:	IdExamenOftalmologicoAnatomia
Descripción:	Propiedad para mostrar y modificar el idExamenOftalmologicoAnatomia
Nombre:	idOtropia
Descripción:	Propiedad para mostrar y modificar el idOtropia

Tabla 14: Clase entidad "ExamenMotilidadOcular"

Nombre: ExamenReflejoPupilar	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenOftalmologicoAnatomia	Int32
idReflejoPupilar	Int32
Para cada responsabilidad:	
Nombre:	ExamenReflejoPupilar
Descripción:	Constructor de la clase.
Nombre:	IdExamenOftalmologicoAnatomia
Descripción:	Propiedad para mostrar y modificar el idExamenOftalmologicoAnatomia
Nombre:	IdReflejoPupilar
Descripción:	Propiedad para mostrar y modificar el idReflejoPupilar

Tabla 15: Clase entidad "ExamenReflejoPupilar"

Nombre: ExamenSegmentoAnterior	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenOftalmologicoAnatomia	Int32
idParámetroSegmentoAnterior	Int32
Para cada responsabilidad:	
Nombre:	ExamenSegmentoAnterior
Descripción:	Constructor de la clase.
Nombre:	IdExamenOftalmologicoAnatomia

Descripción:	Propiedad para mostrar y modificar el idExamenOftalmologicoAnatomia
Nombre:	IdParámetroSegmentoAnterior
Descripción:	Propiedad para mostrar y modificar el idParámetroSegmentoAnterior

Tabla 16: Clase entidad "ExamenSegmentoAnterior"

Nombre: ExamenVitreo	
Tipo de clase: Entidad	
Atributo	**Tipo**
idParámetroVitreo	Int32
idExamenOftalmologicoAnatomia	Int32
Para cada responsabilidad:	
Nombre:	ExamenVitreo
Descripción:	Constructor de la clase.
Nombre:	IdParámetroVitreo
Descripción:	Propiedad para mostrar y modificar el idParámetroVitreo
Nombre:	IdExamenOftalmologicoAnatomia
Descripción:	Propiedad para mostrar y modificar el idExamenOftalmologicoAnatomia

Tabla 17: Clase entidad "ExamenVitreo"

Nombre: ParametroSegmentoAnterior	
Tipo de clase: Entidad	
Atributo	**Tipo**
idParámetroSegmentoAnterior	Int32
descripcion	String

Para cada responsabilidad:	
Nombre:	ParametroSegmentoAnterior
Descripción:	Constructor de la clase.
Nombre:	IdParámetroSegmentoAnterior
Descripción:	Propiedad para mostrar y modificar el idParámetroSegmentoAnterior
Nombre:	Descripcion
Descripción:	Propiedad para mostrar y modificar la descripcion

Tabla 18: Clase entidad "ParametroSegmentoAnterior"

Nombre: AntecedenteOftalmologico	
Tipo de clase: Entidad	
Atributo	**Tipo**
idAntecedenteOftalmologico	Int32
descripcion	String
Para cada responsabilidad:	
Nombre:	AntecedenteOftalmologico
Descripción:	Constructor de la clase.
Nombre:	IdAntecedenteOftalmologico
Descripción:	Propiedad para mostrar y modificar el idAntecedenteOftalmologico
Nombre:	Descripcion
Descripción:	Propiedad para mostrar y modificar la descripcion

Tabla 19: Clase entidad "AntecedenteOftalmologico"

Nombre: HabitoToxico	
Tipo de clase: Entidad	
Atributo	**Tipo**
idHabitoToxico	Int32
descripcion	String
Para cada responsabilidad:	
Nombre:	HabitoToxico
Descripción:	Constructor de la clase.
Nombre:	IdHabitoToxico
Descripción:	Propiedad para mostrar y modificar el idHabitoToxico
Nombre:	Descripcion
Descripción:	Propiedad para mostrar y modificar la descripcion

Tabla 20: Clase entidad "HabitoToxico"

Nombre: Otropia	
Tipo de clase: Entidad	
Atributo	**Tipo**
idOtropia	Int32
descripcion	String
Para cada responsabilidad:	
Nombre:	Otropia
Descripción:	Constructor de la clase.
Nombre:	IdOtropia

Descripción:	Propiedad para mostrar y modificar el idOtropia
Nombre:	Descripcion
Descripción:	Propiedad para mostrar y modificar la descripcion

Tabla 21: Clase entidad "Otropia"

Nombre: Patologia	
Tipo de clase: Entidad	
Atributo	**Tipo**
idPatologia	Int32
Descripción	String
Para cada responsabilidad:	
Nombre:	Patologia
Descripción:	Constructor de la clase.
Nombre:	IdPatologia
Descripción:	Propiedad para mostrar y modificar el idPatologia
Nombre:	Descripcion
Descripción:	Propiedad para mostrar y modificar la descripcion

Tabla 22 Clase entidad "Patologia"

Estas clases fueron creadas con el objetivo de agrupar las entidades mapeadas, creando las relaciones que existen entre ellas. Formando de esta manera las principales entidades del negocio con las que interactúa el sistema.

Nombre: EPRefraccion	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenesOftalmologicos	ExamenOftalmologico
refraccion	Refraccion
Para cada responsabilidad:	
Nombre:	EPRefraccion
Descripción:	Constructor de la clase.
Nombre:	ExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el examenesOftalmologicos
Nombre:	Refraccion
Descripción:	Propiedad para mostrar y modificar la refraccion

Tabla 23: Clase entidad "EPRefraccion"

Nombre: EPQueratometria	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenesOftalmologicos	ExamenOftalmologico
queratometria	Queratometria
Para cada responsabilidad:	
Nombre:	EPQueratometria
Descripción:	Constructor de la clase.

Nombre:	ExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificarel examenesOftalmologicos
Nombre:	Queratometria
Descripción:	Propiedad para mostrar y modificar la queratometria

Tabla 24: Clase entidad "EPQueratometria"

Nombre: EPBiometria	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenesOftalmologicos	ExamenOftalmologico
biometria	Biometria
Para cada responsabilidad:	
Nombre:	EPBiometria
Descripción:	Constructor de la clase.
Nombre:	ExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el examenesOftalmologicos
Nombre:	Biometria
Descripción:	Propiedad para mostrar y modificar la biometria

Tabla 25: Clase entidad "EPBiometria"

Nombre: EPTensionOcular	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenesOftalmologicos	ExamenOftalmologico

tensionOcular	TensionOcular
Para cada responsabilidad:	
Nombre:	EPTensionOcular
Descripción:	Constructor de la clase.
Nombre:	ExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el examenesOftalmologicos
Nombre:	TensionOcular
Descripción:	Propiedad para mostrar y modificar la tensionOcular

Tabla 26: Clase entidad "EPTensionOcular"

Nombre: EPLensometro	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenesOftalmologicos	ExamenOftalmologico
examenLensometro	ExamenLensometro
Para cada responsabilidad:	
Nombre:	EPLensometro
Descripción:	Constructor de la clase.
Nombre:	ExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el examenesOftalmologicos
Nombre:	ExamenLensometro
Descripción:	Propiedad para mostrar y modificar la examenLensometro

Tabla 27: Clase entidad "EPLensometro"

Nombre: EPPaquimetriaCorneal	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenesOftalmologicos	ExamenOftalmologico
paquimetriacorneal	PaquimetriaCorneal
Para cada responsabilidad:	
Nombre:	EPPaquimetriaCorneal
Descripción:	Constructor de la clase.
Nombre:	ExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el examenesOftalmologicos
Nombre:	Paquimetriacorneal
Descripción:	Propiedad para mostrar y modificar la paquimetriacorneal

Tabla 28: Clase entidad "EPPaquimetriaCorneal"

Nombre: EPRefraccionIndicada	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenesOftalmologicos	ExamenOftalmologico
examenRefraccionIndicada	ExamenRefraccionIndicada
Para cada responsabilidad:	
Nombre:	EPRefraccionIndicada
Descripción:	Constructor de la clase.
Nombre:	ExamenesOftalmologicos

Descripción:	Propiedad para mostrar y modificar el examenesOftalmologicos
Nombre:	ExamenRefraccionIndicada
Descripción:	Propiedad para mostrar y modificar la examenRefraccionIndicada

Tabla 29: Clase entidad "EPRefraccionIndicada"

Nombre: EPRefraccionDinamica	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenesOftalmologicos	ExamenOftalmologico
examenRefraccionDinamica	ExamenRefraccionDinamica
Para cada responsabilidad:	
Nombre:	EPRefraccionIndicada
Descripción:	Constructor de la clase.
Nombre:	ExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar el examenesOftalmologicos
Nombre:	ExamenRefraccionDinamica
Descripción:	Propiedad para mostrar y modificar la examenRefraccionDinamica

Tabla 30: Clase entidad "EPRefraccionDinamica"

2.6.2 Clases Controladoras.

Nombre: RefraccionRepositorio	
Tipo de clase: Controladora	
Atributo	**Tipo**
factory	IDomainObjectFactory<Refraccion>
Para cada responsabilidad:	
Nombre:	RefraccionRepositorio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno
Descripción:	Requiere de un parámetro de tipo Refraccion y devuelve la consulta especializada oftalmológica correspondiante a los datos entrados.
Nombre:	ObtenerTodos
Descripción:	Requiere de un parámetro de tipo Refraccion y devuelve una lista de consultas especializada oftalmológica correspondiantes a los datos entrados.
Nombre:	Adicionar
Descripción:	Requiere de un parámetro de tipo Refraccion, el cual sera guardado en base de datos.
Nombre:	Actualizar
Descripción:	Requiere de un parámetro de tipo Refraccion, el cual sera actualizado en base de datos.
Nombre:	Eliminar
Descripción:	Requiere de un parámetro de tipo Refraccion, el cual sera suprimido de base de datos.

Tabla 31: Clase controladora “RefraccionRepositorio”

Nombre: RefraccionSelectionFactory	
Tipo de clase:	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	RefraccionSelectionFactory
Descripción:	Constructor de la clase.
Nombre:	Execute
Descripción:	Devuelve una lista de Refraccion, requiere de tres parámetros el helper, el domainObjectFactory y la entidad.

Tabla 32: Clase controladora "RefraccionSelectionFactory"

Nombre: RefraccionInsertFactory	
Tipo de clase:	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	RefraccionInsertFactory
Descripción:	Constructor de la clase.
Nombre:	Execute
Descripción:	Inserta en la BD un objeto de tipo Refraccion, requiere del parámetro helper.

Tabla 33: Clase controladora "RefraccionInsertFactory"

Nombre: RefraccionUpdateFactory	
Tipo de clase:	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	RefraccionUpdateFactory
Descripción:	Constructor de la clase
Nombre:	Execute
Descripción:	Actualiza en la BD el objeto de tipo Refraccion pasado por parámetro, requiere del helper

Tabla 34: Clase controladora "RefraccionUpdateFactory"

Nombre: RefraccionDeleteFactory	
Tipo de clase:	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	RefraccionDeleteFactory
Descripción:	Constructor de la clase.
Nombre:	Execute
Descripción:	Suprime de la BD el objeto de tipo Refraccion pasado por parámetro, requiere del helper.

Tabla 35: Clase controladora "RefraccionDeleteFactory"

Nombre: EPRefraccionNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
refraccionRepositorio	EPRefraccionRepositorio
Para cada responsabilidad:	
Nombre:	EPRefraccionNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EPrefraccion*)
Descripción:	Método que devuelve la consulta especializada oftalmológica correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EPrefraccion*)
Descripción:	Método que devuelve una lista de consultas especializadas oftalmológicas correspondiante a los datos entrados.
Nombre:	Adicionar(*EPrefraccion*)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EPrefraccion*)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EPrefraccion*)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 36: Clase controladora "EPRefraccionNegocio"

Nombre: EPQueratometriaNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
queratometriaRepositorio	EPQueratometriaRepositorio
Para cada responsabilidad:	
Nombre:	EPQueratometriaNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EPqueratometria*)
Descripción:	Método que devuelve la consulta especializada oftalmológica correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EPqueratometria*)
Descripción:	Método que devuelve una lista de consultas especializadas oftalmológicas correspondiante a los datos entrados.
Nombre:	Adicionar(*EPqueratometria*)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EPqueratometria*)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EPqueratometria*)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 37: Clase controladora "EPQueratometriaNegocio"

Nombre: EPBiometriaNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
biometriaRepositorio	EPBiometriaRepositorio
Para cada responsabilidad:	
Nombre:	EPBiometriaNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EP*biometria)
Descripción:	Método que devuelve la consulta especializada oftalmológica correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EP*biometria)
Descripción:	Método que devuelve una lista de consultas especializadas oftalmológicas correspondiante a los datos entrados.
Nombre:	Adicionar(*EP*biometria)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EP*biometria)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EP*biometria)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 38: Clase controladora "EPBiometriaNegocio".

Nombre: EPTensionOcularNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
tensionOcularRepositorio	EPTensionOcularRepositorio
Para cada responsabilidad:	
Nombre:	EPTensionOcularNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EP*tensionOcular)
Descripción:	Método que devuelve la consulta especializada oftalmológica correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EP*tensionOcular)
Descripción:	Método que devuelve una lista de consultas especializadas oftalmológicas correspondiante a los datos entrados.
Nombre:	Adicionar(*EP*tensionOcular)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EP*tensionOcular)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EP*tensionOcular)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 39: Clase controladora "EPTensionOcularNegocio".

Nombre: EPLensometroNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
lensometroRepositorio	EPLensometroRepositorio
Para cada responsabilidad:	
Nombre:	EPLensometroNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EP*lensometro)
Descripción:	Método que devuelve la consulta especializada oftalmológica correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EP*lensometro)
Descripción:	Método que devuelve una lista de consultas especializadas oftalmológicas correspondiante a los datos entrados.
Nombre:	Adicionar(*EP*lensometro)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EP*lensometro)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EP*lensometro)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 40: Clase controladora "EPLensometroNegocio".

Nombre: EPRefraccionIndicadaNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
refraccionIndicadaRepositorio	EPRefraccionIndicadaRepositorio
Para cada responsabilidad:	
Nombre:	EPRefraccionIndicadaNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EP*refraccionIndicada)
Descripción:	Método que devuelve la consulta especializada oftalmológica correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EP*refraccionIndicada)
Descripción:	Método que devuelve una lista de consultas especializadas oftalmológicas correspondiante a los datos entrados.
Nombre:	Adicionar(*EP*refraccionIndicada)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EP*refraccionIndicada)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EP*refraccionIndicada)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 41: Clase controladora "EPRefraccionIndicadaNegocio".

Nombre: EPRefraccionDinamicaNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
refraccionDinamicaRepositorio	EPRefraccionDinamicaRepositorio
Para cada responsabilidad:	
Nombre:	EPRefraccionDinamicaNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EP*refraccionDinamica)
Descripción:	Método que devuelve la consulta especializada oftalmológica correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EP*refraccionDinamica)
Descripción:	Método que devuelve una lista de consultas especializadas oftalmológicas correspondiante a los datos entrados.
Nombre:	Adicionar(*EP*refraccionDinamica)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EP*refraccionDinamica)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EP*refraccionDinamica)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 42: Clase controladora "EPRefraccionIndicadaNegocio".

Nombre: EPPaquimetriaCornealNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
paquimetriaCornealRepositorio	EPPaquimetriaCornealRepositorio
Para cada responsabilidad:	
Nombre:	EPPaquimetriaCornealNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EP*paquimetria)
Descripción:	Método que devuelve la consulta especializada oftalmológica correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EP*paquimetria)
Descripción:	Método que devuelve una lista de consultas especializadas oftalmológicas correspondiante a los datos entrados.
Nombre:	Adicionar(*EP*paquimetria)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EP*paquimetria)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EP*paquimetria)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 43: Clase controladora "EPPaquimetriaCornealNegocio".

2.6.3 Clases interfaces.

Nombre: bqo_CirugiaRefractiva	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSListConsulta	List<EPConsulta>
SSConsulta	EPConsultaEspecializadaOftalmologica
ControGuardar	Int32
ModificarConsulta	Boolean
PuedeCrearAnuncio	Boolean
SSDatosPersonas	DatosPersonas
SSEPAnuncioOperatorioGeneral	EPAnuncioOperatorioGeneral
autorizacion	SAAAAutorizaWSDL
SSTokens	Tokens
Para cada responsabilidad:	
Nombre:	Page_Load
Descripción:	Evento que se ejecuta cuando se carga la página. En el se inicializan las listas de los nomencladores a usar en esta página.
Nombre:	SeleccionarConsultaHistorial
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:	VerificarAnuncio
Descripción:	Método para verificar si el médico puede crear anuncios operatorios correspondientes a la consulta seleccionada.
Nombre:	GuardarDatos
Descripción:	Método para guardar toda la información recogida durante la consulta en una variable de sesión.
Nombre:	ControlesGuardados
Descripción:	Método para verificar si todos los controles fueron guardados satisfactoriamente.
Nombre:	RefrescarAnuncio
Descripción:	Método para mostrar nuevamente la lista de anuncios operatorios correspondientes a la consulta seleccionada.
Nombre:	InhabilitarControles
Descripción:	Método para inhabilitar los controles cuando haya pasado el tiempo de modificación de la consulta.
Nombre:	ButtonTab_Click
Descripción:	Evento que se ejecuta sobre los botones del tab. En este evento se activa el tab correspondiente al boton pulsado
Nombre:	Button_Click
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:	GridViewConsultas_RowCreated
Descripción:	Evento que se ejecuta cuando el gridview crea cada una de las filas. En el se le

	especifica como debe crear esta fila.
Nombre:	GridViewConsultas_PageIndexChanging
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
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:	MultiViewConsulta_ActiveViewChanged
Descripción:	Evento que se ejecuta al cambiar de tab. En el se guardan los datos del tab que estaba activo.
Nombre:	GridViewAnuncio_RowCreated
Descripción:	Evento que se ejecuta cuando el gridview crea cada una de las filas. En el se le especifica como debe crear esta fila.
Nombre:	ButoonExamenesEsp_Click
Descripción:	Evento que se ejecuata al hacer click en el butoon Examenes especializados que redirecciona a nueva página donde se encuantran los exámenes especializados de la consulta en cuestión.

Tabla 44: Clase interfaz "bqo_CirugiaRefractiva"

Nombre: bqo_CirugiaCristalino	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSListConsulta	List<EPConsulta>
SSConsulta	EPConsultaEspecializadaOftalmologica
ControGuardar	Int32
ModificarConsulta	Boolean

PuedeCrearAnuncio	Boolean
SSDatosPersonas	DatosPersonas
SSEPAnuncioOperatorioGeneral	EPAnuncioOperatorioGeneral
autorizacion	SAAAAutorizaWSDL
SSTokens	Tokens
Para cada responsabilidad:	
Nombre:	Page_Load
Descripción:	Evento que se ejecuta cuando se carga la página. En el se inicializan las listas de los nomencladores a usar en esta página.
Nombre:	SeleccionarConsultaHistorial
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:	VerificarAnuncio
Descripción:	Método para verificar si el médico puede crear anuncios operatorios correspondientes a la consulta seleccionada.
Nombre:	GuardarDatos
Descripción:	Método para guardar toda la información recogida durante la consulta en una variable de sesión.
Nombre:	ControlesGuardados

Descripción:	Método para verificar si todos los controles fueron guardados satisfactoriamente.
Nombre:	RefrescarAnuncio
Descripción:	Método para mostrar nuevamente la lista de anuncios operatorios correspondientes a la consulta seleccionada.
Nombre:	InhabilitarControles
Descripción:	Método para inhabilitar los controles cuando haya pasado el tiempo de modificación de la consulta.
Nombre:	ButtonTab_Click
Descripción:	Evento que se ejecuta sobre los botones del tab. En este evento se activa el tab correspondiente al boton pulsado
Nombre:	Button_Click
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:	GridViewConsultas_RowCreated
Descripción:	Evento que se ejecuta cuando el gridview crea cada una de las filas. En el se le especifica como debe crear esta fila.
Nombre:	GridViewConsultas_PageIndexChanging
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
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:	MultiViewConsulta_ActiveViewChanged
Descripción:	Evento que se ejecuta al cambiar de tab. En el se guardan los datos del tab que

	estaba activo.
Nombre:	GridViewAnuncio_RowCreated
Descripción:	Evento que se ejecuta cuando el gridview crea cada una de las filas. En el se le especifica como debe crear esta fila.
Nombre:	Butoon2_Click
Descripción:	Evento que se ejecuata al hacer click en el butoon Examenes especializados que redirecciona a nueva página donde se encuantran los exámenes especializados de la consulta en cuestión.

Tabla 45: Clase interfaz "bqo_CirugiaCristalino"

Nombre: bqo_CirugiaPterigium	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSListConsulta	List<EPConsulta>
SSConsulta	EPConsultaEspecializadaOftalmologica
ControGuardar	Int32
ModificarConsulta	Boolean
PuedeCrearAnuncio	Boolean
SSDatosPersonas	DatosPersonas
SSEPAnuncioOperatorioGeneral	EPAnuncioOperatorioGeneral
autorizacion	SAAAAutorizaWSDL
SSTokens	Tokens
Para cada responsabilidad:	
Nombre:	Page_Load
Descripción:	Evento que se ejecuta cuando se carga la página. En el se inicializan las listas de los

	nomencladores a usar en esta página.
Nombre:	SeleccionarConsultaHistorial
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:	VerificarAnuncio
Descripción:	Método para verificar si el médico puede crear anuncios operatorios correspondientes a la consulta seleccionada.
Nombre:	GuardarDatos
Descripción:	Método para guardar toda la información recogida durante la consulta en una variable de sesión.
Nombre:	ControlesGuardados
Descripción:	Método para verificar si todos los controles fueron guardados satisfactoriamente.
Nombre:	RefrescarAnuncio
Descripción:	Método para mostrar nuevamente la lista de anuncios operatorios correspondientes a la consulta seleccionada.
Nombre:	InhabilitarControles
Descripción:	Método para inhabilitar los controles cuando haya pasado el tiempo de modificación de la consulta.
Nombre:	ButtonTab_Click
Descripción:	Evento que se ejecuta sobre los botones del tab. En este evento se activa el tab correspondiente al boton pulsado

Nombre:	Button_Click
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:	GridViewConsultas_RowCreated
Descripción:	Evento que se ejecuta cuando el gridview crea cada una de las filas. En el se le especifica como debe crear esta fila.
Nombre:	GridViewConsultas_PageIndexChanging
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
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:	MultiViewConsulta_ActiveViewChanged
Descripción:	Evento que se ejecuta al cambiar de tab. En el se guardan los datos del tab que estaba activo.
Nombre:	GridViewAnuncio_RowCreated
Descripción:	Evento que se ejecuta cuando el gridview crea cada una de las filas. En el se le especifica como debe crear esta fila.

Tabla 46: Clase interfaz "bqo_CirugiaPterigium"

Nombre: ExamenesEspCristalino	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSDatosPersonas	DatosPersonas

ControlGuardar	Int32
VSepRefraccion	EPRefraccion
aa	EPRefraccion
examenBiometria	EPBiometria
examenQueratometria	EPQueratometria
examenTensionOcular	EPTensionOcular
VSlistaExamenes	List<EPRefraccion>
VSlistaExamenBiometria	List<EPBiometria>
VSlistaExamenQueratometria	List<EPQueratometria>
VSlistaTensionOcular	List<EPTensionOcular>
VSBandera	Int32
fechaExamen	DateTime
culture	System
autorizacion	SAAAAutorizaWSDL
SSTokens	Tokens
Para cada responsabilidad:	
Nombre:	Page_Load
Descripción:	Evento que se ejecuta cuando se carga la página. En el se inicializan las listas de los nomencladores a usar en esta página.
Nombre:	GridViewAnuncio_RowCreated
Descripción:	Evento que se ejecuta cuando el GridView crea cada una de las filas. En el se le especifica como debe crear esta fila.
Nombre:	GridViewAnuncio_SelectedIndexChanging
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:	Click
Descripción:	Evento que se ejecuta al hacer click sobre los botones del multiview. Según el botón que fue presionado se ejecutaran acciones como: Mostrar el control web que le corresponde de acuerdo con el Index del butoon.
Nombre:	ButtonTab_Click
Descripción:	Evento que se ejecuta sobre los botones del tab. En este evento se activa el tab correspondiente al boton pulsado
Nombre:	GridView_PageIndexChanging
Descripción:	Evento que se ejecuta al hacer click en el paginado del gridview. En el se le especifica como debe hacer el paginado.
Nombre:	ButoonCabecera1_Click
Descripción:	Evento que se ejecuta al hacer click sobre los botones de la cabecera. Según el botón que fue presionado se ejecutaran acciones como: Nuevo examen. Guardar todos los controles.

Tabla 47: Clase interfaz "ExamenEspCristalino"

Nombre: ExamenesEspRefractiva	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSDatosPersonas	DatosPersonas
ControlGuardar	Int32
VSepRefraccion	EPRefraccion
aa	EPRefraccion

examenBiometria	EPBiometria
examenQueratometria	EPQueratometria
examenTensionOcular	EPTensionOcular
examenLensometro	EPLensometro
examenPaquimetria	EPPaquimetriaCorneal
examenRefraccionIndicada	EPRefraccionIndicada
examenRefraccionDinamica	EPRefraccionDinamica
VSlistaExamenes	List<EPRefraccion>
VSlistaExamenBiometria	List<EPBiometria>
VSlistaExamenQueratometria	List<EPQueratometria>
VSlistaTensionOcular	List<EPTensionOcular>
VSExamenLensometro	List<EPLensometro>
VSExamenesPaquimetriaCorneal	List<EPPaquimetriaCorneal>
VSListarefraccionIndicada	List<EPRefraccionIndicada>
VSListarefraccionDinamica	List<EPRefraccionDinamica>
VSBandera	Int32
fechaExamen	DateTime
culture	System
autorizacion	SAAAAutorizaWSDL
SSTokens	Tokens
Para cada responsabilidad:	
Nombre:	Page_Load
Descripción:	Evento que se ejecuta cuando se carga la página. En el se inicializan las listas de los

	nomencladores a usar en esta página.
Nombre:	GridViewAnuncio_RowCreated
Descripción:	Evento que se ejecuta cuando el GridView crea cada una de las filas. En el se le especifica como debe crear esta fila.
Nombre:	GridViewAnuncio_SelectedIndexChanging
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:	Click
Descripción:	Evento que se ejecuta al hacer click sobre los botones del multiview. Según el botón que fue presionado se ejecutaran acciones como: Mostrar el control web que le corresponde de acuerdo con el Index del butoon.
Nombre:	ButtonTab_Click
Descripción:	Evento que se ejecuta sobre los botones del tab. En este evento se activa el tab correspondiente al boton pulsado
Nombre:	GridView_PageIndexChanging
Descripción:	Evento que se ejecuta al hacer click en el paginado del gridview. En el se le especifica como debe hacer el paginado.
Nombre:	ButoonCabecera1_Click
Descripción:	Evento que se ejecuta al hacer click sobre los botones de la cabecera. Según el botón que fue presionado se ejecutaran acciones como: Nuevo examen. Guardar todos los controles.

Tabla 48: Clase interfaz **"ExamenEspRefractiva"**

Nombre: GestionarUsuario	
Tipo de clase: Interfaz	
Atributo	**Tipo**
listarUsuario	SAAAListarUsuariosWSDL.Listarusuarios
listarUserRetorno	SAAAListarUsuariosWSDL.ListarUsuariosRetorno
listarUserResponse	SAAAListarUsuariosWSDL.ListarusuarioResponse
usuarioSAAA	SAAAListarUsuariosWSDL.Usuario
derechoUser	SAAAListarUsuariosWSDL.derechousuario
arregloIdArregloIdItem	SAAAListarUsuariosWSDL.ArregloIdArregloIdItem
arregloMod	SAAAListarUsuariosWSDL.ArregloIdArregloIdItem[12]
arregloIteUser	SAAAListarUsuariosWSDL.ArregloIdArregloIdItem[21]
usuarioDatos	UsuarioCorto
arregloUsuariosDatos	UsuarioCorto[]
listaUserData	List<UsuarioCorto>
bandera	int
eliminarUsuario	SAAAEliminarUsuarioWSDL.eliminarusuario
eliminarUserResponse	SAAAEliminarUsuarioWSDL.Response
listarModulo	SAAAListarModulosWSDL.listarmodulos
ListarModuloResp	SAAAListarModulosWSDL.ListarModulosRetorno
arregloIte	SAAAListarModulosWSDL.ArregloIdArregloIdItem[]
autorizacion	SAAAAutorizaWSDL
SSTokens	Tokens
Para cada responsabilidad:	

Nombre:	Page_Load
Descripción:	Evento que se ejecuta cuando se carga la página. En el se inicializan las listas de los nomencladores a usar en esta página.
Nombre:	GridViewUsuarios_RowDeleting
Descripción:	Evento que se ejecuta cuando en el GridViewUsuarios se presiona el botón "eliminar" de determinada fila, se eliminará el usuario de dicha fila y el GridViewUsuarios cargará nuevamente la lista de usuarios modificada.
Nombre:	GridViewUsuarios_RowEditing
Descripción:	Evento que se ejecuta cuando en el GridViewUsuarios se presiona el botón "Editar" de determinada fila, se redirecciona a la página "ModificarUsuario ", donde se cargarán todos los valoresde dicho usuario y te dará la posibilidad de cambiarlos.
Nombre:	DropDownListModulos_SelectedIndexChanged
Descripción:	Evento que se ejecuta al seleccionar un Módulo del DropDownListModulos, seguidamente se ejecutaran acciones como: El GridViewUsuarios carga los usuarios que tienen permiso sobre ese Módulo.
Nombre:	AcortarUsuarios
Descripción:	Método auxiliar que recibe una lista de usarios de tipo (*SAAAListarUsuariosWSDL.Usuario*) , y los transforma en el tipo(*UsuarioCorto*) , devolviendolos en una lista de estos últimos.
Nombre:	BuscarPorNombre
Descripción:	Método auxiliar que recibe un *string*, y busca los usuarios que tengan como nombre el *string* pasado como parámetro y lo devuelve en una lista *List<UsuarioCorto>.*
Nombre:	DropDownListCodigouser_SelectedIndexChanged
Descripción:	Evento que se ejecuta al seleccionar un id de los cargados en el DropDownListCodigouser. Según el id que fue seleccionado, se ejecutaran acciones como: . El GridViewUsuarios cargará el usuario que tiene el mismo id que el seleccionado.

Nombre:	ButtonNuevoUser_Click
Descripción:	Evento que se ejecuta al hacer ClicK sobre el botón "*Nuevo Usuario*", inmediatamente será redireccionado a la página "*AgregarUsuario*".
Nombre:	ButtonNombreUser_Click
Descripción:	Evento que se ejecuta al hacer ClicK sobre el botón "*ButtonNombreUse*r", se mostrará el textBox "*TextBoxNombreBusc*" y el botón "*ButtonBuscarUser*".
Nombre:	ButtonModuloUser_Click
Descripción:	Evento que se ejecuta al hacer ClicK sobre el botón "*ButtonModuloUser*", se caragará el DropDownList, "*DropDownListModulos*", con todos los módulos del sistema.
Nombre:	ButtonIdUsuer_Click
Descripción:	Evento que se ejecuta al hacer ClicK sobre el botón "*ButtonIdUsuer*", se caragará el DropDownList, "*DropDownListCodigouser*", con todos los ID de los usuarios del sistema.
Nombre:	ButtonBuscarUser_Click
Descripción:	Evento que se ejecuta al hacer ClicK sobre el botón" ButtonBuscarUser", invocará a la función "BuscarPorNombre" , recibiendo como parámetro el texto del textBox "*TextBoxNombreBusc*", cargando así el gridView "*GridViewUsuarios*" con los usuarios que cumpla con el parámetro pasado.

Tabla 49: Clase interfaz "GestionarUsuario"

Nombre: AgregarUsuario	
Tipo de clase: Interfaz	
Atributo	**Tipo**
listarUsuario	SAAAListarUsuariosWSDL.Listarusuarios
listarUserRetorno	SAAAListarUsuariosWSDL.ListarUsuariosRetorno
listarUserResponse	SAAAListarUsuariosWSDL.ListarusuarioResponse

usuarioSAAA	SAAAListarUsuariosWSDL.Usuario
arregloIdArregloIdItem	SAAAListarUsuariosWSDL.ArregloIdArregloIdItem
arregloID	SAAAListarUsuariosWSDL.ArregloIdArregloIdItem[]
ListaAux	List<SAAARegistrarUsuarioWSDL.itemusuario>
ListaModuloPermiso	List<ModuloPermiso>
itemModulo1	SAAAListarModulosWSDL.itemModulo
derechos	SAAAListarModulosWSDL.DerechoDatos[]
derechosComponente	SAAAListarModulosWSDL.DerechoDatos[]
listarModulo	SAAAListarModulosWSDL.listarmodulos
ListarModuloResp	SAAAListarModulosWSDL.ListarModulosRetorno
arregloIte	SAAAListarModulosWSDL.ArregloIdArregloIdItem[]
modificarUsuario	SAAAModificarUsuarioWSDL.ModificarUsuario
modificarUserReq	SAAAModificarUsuarioWSDL.registrarusuarioReques
modificarUserResponse	SAAAModificarUsuarioWSDL.Response
modificarUserItem	SAAAModificarUsuarioWSDL.itemusuario
ModificarDerechosUser	SAAAModificarUsuarioWSDL.itemusuario[]
ListaModificarDerechos	List<SAAAModificarUsuarioWSDL.itemusuario>
Bandera	int
actualiza	int
autorizacion	SAAAAutorizaWSDL
SSTokens	Tokens
Para cada responsabilidad:	
Nombre:	Page_Load

Descripción:	Evento que se ejecuta cuando se carga la página. En el se inicializan las listas de los nomencladores a usar en esta página.
Nombre:	GridViewModuloPermiso_RowDeleting
Descripción:	Evento que se ejecuta cuando en el gridView "GridViewModuloPermiso" se presiona el botón "eliminar" de determinada fila, se eliminará el módulo de dicha fila y el el gridView "GridViewModuloPermiso" cargará nuevamente la lista de módulos modificada.
Nombre:	DropDownListModulos_SelectedIndexChanged
Descripción:	Evento que se ejecuta al seleccionar un Módulo del DropDownListModulos, seguidamente se ejecutaran acciones como: El DropDownList "*DropDownListDErechosModulo*" carga los derechos que posee el módulo seleccionado.
Nombre:	VerificarModuloNewUser(*short* id)
Descripción:	Método auxiliar que recibe un *short(id del módulo),* verifica si a un usuario determinado se la asignó anterirormente un módulo con ese id, de encontrarlo retormanrá *true,* de lo contrario retornará *false.*
Nombre:	VerificarModulo(*short* id)
Descripción:	Método auxiliar que recibe un *short(id del módulo)*, y busca si existe un módulo registrado con el id recibido, de encontrarlo retornará *true* de lo contrario retornará *false*.
Nombre:	ConvertirMinusculas(*string* cadena)
Descripción:	Método auxiliar que recibe un *string* como parámetro, y a partir de la segunda palabra las convertirá en minúsculas todas las restantes, devolverá otro *string* con los cambios efectuados.
Nombre:	Button1_Click1
Descripción:	Evento que se ejecuta al hacer ClicK sobre el botón "Button1", seguidamente se ejecutaran acciones como: Se verifica si se está adicionado un nuevo usuario y precede a adicionarlo; de lo contrario, estaría modificando un usuario y realizaría dicha acción("Modificar

	Usuario").
Nombre:	ButtonListar_Click
Descripción:	Evento que se ejecuta al hacer ClicK sobre el botón "ButtonListar", seguidamente se ejecutaran acciones como: El gridView "GridViewModuloPermiso" cargará un nuevo módulo con el permiso seleccionado para asignarselo al nuevo usuario.
Nombre:	CargarUsuarioMod()
Descripción:	Método auxiliar que será invocado desde el Page_Load, después de ser verificado que la acción a realizar en la página es "Modificar un Usuario". Este método cargará todos los datos archivados del usuario seleccionado para su posible modificación.
Nombre:	BuscarModulo(*short* id)
Descripción:	Método auxiliar al que se le pasará un id como parámetro, y devolverá el módulo que tenga el id recibido, de no encontrarse nigún módulo con el id devolverá *null*.
Nombre:	BuscarDerecho(*short* id)
Descripción:	Método auxiliar al que se le pasará un id como parámetro, y devolverá el nombre del módulo que tenga el id recibido, de no encontrarse nigún módulo con el id devolverá *null.*

Tabla 50: Clase interfaz **"AgregarUsuario"**

Nombre: AgregarMetodo	
Tipo de clase: Interfaz	
Atributo	**Tipo**
nuevoMetodo	SAAANuevoMetodoWSDL.Nuevometodo
registraMetodo	SAAANuevoMetodoWSDL.registrarmetodo
derechoCodigo	SAAANuevoMetodoWSDL.DerechoCodigo
metResp	SAAANuevoMetodoWSDL.Response
arregloDer	SAAANuevoMetodoWSDL.DerechoCodigo[]

arregloNiv	SAAANuevoMetodoWSDL.ArregloNivel
arregloNivel1	SAAANuevoMetodoWSDL.ArregloNivel[]
derechos	SAAAListarModulosWSDL.DerechoDatos[]
derechosComponente	SAAAListarModulosWSDL.DerechoDatos[]
listarModulo	SAAAListarModulosWSDL.listarmodulos
ListarModuloResp	SAAAListarModulosWSDL.ListarModulosRetorno
arregloIte	SAAAListarModulosWSDL.ArregloIdArregloIdItem[]
autorizacion	SAAAAutorizaWSDL
SSTokens	Tokens
Para cada responsabilidad:	
Nombre:	Page_Load
Descripción:	Evento que se ejecuta cuando se carga la página. En el se inicializan las listas de los nomencladores a usar en esta página.
Nombre:	ButtonNuevoMetodo_Click
Descripción:	Evento que se ejecuta al presionar sobre el botón “ButtonNuevoMetodo”, seguidamente se asiganarán todos los valores introducidos por el usuario y se agregará un nuevo método al sistema.
Nombre:	DropDownListModulos_SelectedIndexChanged
Descripción:	Evento que se ejecuta al seleccionar un Módulo del DropDownListModulos, seguidamente se ejecutaran acciones como: El gridView “GridViewDerechosModulo” cargará todos los derechos del módulo seleccionado.

Tabla 51: Clase interfaz “AgregarMetodo”

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

En este capítulo se analizaron las posibles implementaciones de los servicios oftalmológicos Pterigium, Cirugía del cristalino, Cirugía refractiva y los módulos Administración y Configuración. Se describieron las clases, los tipos de datos y las soluciones que se implementan para dar respuesta al problema, se realizo el cálculo para uno de los algoritmos no triviales del sistema llegando a la conclusión que la evaluación de riesgo de la aplicación demuestra que es un programa simple, sin mucho riesgo.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

La prueba del software es un elemento crítico para la garantía de la calidad del mismo. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Esta etapa implica:

- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

La prueba no es una actividad sencilla, no es una etapa del proyecto en la cual se asegura la calidad, sino que la prueba debe ocurrir durante todo el ciclo de vida: se puede probar la funcionalidad de los primeros prototipos; probar la estabilidad, cobertura y rendimiento de la arquitectura; probar el producto final. Lo que conduce al principal beneficio de la prueba: proporcionar feedback mientras hay todavía tiempo y recursos para hacer algo.

La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. La prueba es un proceso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

La prueba ideal de un sistema sería exponerlo en todas las situaciones posibles, así se encontraría hasta el último fallo. Indirectamente, se garantiza su respuesta ante cualquier caso que se le presente en la ejecución real.

Esto es imposible desde todos los puntos de vista: humano, económico e incluso matemático. Dado que todo es finito en programación (el número de líneas de código, el número de variables, el número de valores en un tipo) cabe pensar que el número de pruebas posibles es finito. Esto deja de ser cierto en cuanto entran en juego bucles, en los que es fácil introducir condiciones para un funcionamiento sin fin. Aún en el irrealista caso de que el número de posibilidades fuera finito, el número de combinaciones posibles es tan enorme que se hace imposible su identificación y ejecución a todos los efectos prácticos.

3.1 Pruebas a realizar en tiempo de desarrollo.

Normalmente cabe distinguir una fase informal antes de entrar en la fase de pruebas propiamente dicha. La fase informal la lleva a cabo el propio programador en su puesto de trabajo, y consiste en ir ejecutando el código para convencerse de que "básicamente, funciona". Esta fase suele consistir en pequeños ejemplos que se intentan ejecutar. Si el módulo falla, se suele utilizar un depurador para observar la evolución dinámica del sistema, localizar el fallo, y repararlo.

Dentro de las pruebas en tiempo de desarrollo se encuentran las *pruebas de unidades*, estas son pruebas de menor escala y consisten en probar cada uno de los módulos que conforman el programa, cuando estos módulos son extensos o complejos se dividen para probar objetivamente partes más pequeñas, este tipo de pruebas es la más común.

Las *pruebas de integración* tienen por objetivo verificar el conjunto funcionamiento de dos o más módulos, si bien se deben poner en práctica desde la creación de dos módulos que interactúen entre si, en el supuesto caso que se necesiten más de dos módulos para efectuar las pruebas, deberán generarse simples emuladores de módulos que entreguen datos esperados para la prueba individual de cada uno.

También las pruebas de integración pueden ser realizadas en forma ascendente, esto evita tener que crear módulos emuladores, ya que a medida que se va creando la pirámide va siendo probada de abajo hacia arriba (*Down to Top*), como se imaginaran esto acarrea un trabajo simétricamente mayor lo que equipara o supera el tiempo que podría tomar el crear módulos para prueba.[27]

3.2 Pruebas después de la programación.

Cuando se considera que un módulo está terminado se realizan las *pruebas sistemáticas*, el objetivo de estas es buscar fallos a través de un criterio específico, estos criterios se denominan "pruebas de caja negra y de caja blanca".

Las *pruebas de caja negra* son aquellas que se enfocan directamente en el exterior del módulo, sin importar el código, son pruebas funcionales en las que se trata de encontrar fallas en las que este no se atiene a su especificación, como ser interfaz con el usuario, apariencia de los menús, control de las teclas.

Este tipo de pruebas no es aplicable a los módulos que trabajan en forma transparente al usuario.

Para realizar estas pruebas existe una técnica algebraica llamada "clases de equivalencia", consiste en tratar a todas las posibles entradas y parámetros como un modelo algebraico, y utilizar las clases de este modelo para probar un amplio rango de posibilidades.

Para la generación de estas clases no se puede armar un modelo, pero se pueden seguir las siguientes pautas como guía utilizable para la creación de cada clase.

Por ejemplo:

Cuando una entrada es booleana, existen solo dos clases, verdadero o falso.

Las *pruebas de caja blanca* son mucho más amplias, normalmente se denominan pruebas de cobertura o pruebas de caja transparente, al total de pruebas de caja blanca se le llama cobertura, la cobertura es un número porcentual que indica cuanto código del programa se ha probado.

Básicamente la idea de pruebas de cobertura consiste en diseñar un plan de pruebas en las que se vaya ejecutando sistemáticamente el código hasta que haya corrido todo o la gran mayoría de él, esto que parece complicado es más aún cuando el programa contiene código de difícil alcance, como por ejemplo manejadores de errores o "código muerto".

Entiéndase por código muerto aquellas funciones y procedimientos que se han incluido por encontrarse en las recopilaciones, pero nunca son ejecutadas por el programa. Estas funciones no necesariamente deberán ser removidas; pero si probadas por si en revisiones futuras son incluidas.

Pruebas de aceptación, son las que hará el cliente, en esta fase de pruebas se determina que el sistema cumple con el objetivo deseado, determina la conformidad del cliente antes de que el programa le sea entregado como una versión final.

Las pruebas conocidas con el nombre de *pruebas de rendimiento* son aquellas que determinan los tiempos de respuesta, el espacio que ocupa el módulo en disco o en memoria, el flujo de datos que genera a través de un canal de comunicaciones.

Pruebas de transformación, este método curioso y caro aún se pone en funcionamiento por diversas empresas, consiste en dividir el equipo de desarrollo en dos partes una vez realizadas todas las pruebas y corregidos todos los errores, luego una de las dos partes introduce pequeños errores en el sistema y la otra parte debe encontrarlos con los mismos procedimientos que se usaron para buscar los errores nativos. Esto es muy costoso y consume grandes cantidades de tiempo.

Pruebas de robustez, comúnmente denominadas "*robustness test*" son las encargadas de verificar la capacidad del programa para soportar entradas incorrectas, por ejemplo en un sistema de facturación

donde el usuario debe ingresar códigos de productos y luego cantidades es más que factible que en algún momento ingrese un código en el campo de cantidad, si el programa fue sometido a pruebas de robustez este valor sería rechazado o grabado como una cantidad inmensa pero que no daría error por desbordamiento de datos.

Las denominadas *pruebas de resistencia* se utilizan para saber hasta dónde puede soportar el programa condiciones extremas, por ejemplo los tiempos de respuesta con el procesador a un 95% de su utilidad o con muy poco espacio en disco.[27]

3.3 Pruebas de caja blanca.

Para realizar las pruebas de caja blanca se siguen los pasos descritos en Capítulo 2. Epígrafe 2.4.1 Análisis de complejidad.

Código para insertar el ojo derecho en el examen de Refraccion de la cosulta cirugía Cristalino.

```
protected void GuardarOjoDerecho()
  {    1
       ExamenOD(); 1
        if (GuardadoOD) 2
        {

           new EPRefraccionNegocio().Adicionar(SSExamenRefraccionOD); 3
           ButtonGuardar.Enabled = true; 3
           LabelError.Text = "Registro insertado satisfactoriamente"; 3
           LabelError.Font.Bold = true; 3
           LabelError.ForeColor = System.Drawing.Color.Teal; 3
           ButtonGuardar.Enabled = false; 3
           Label3.Text = ""; 3
        }
        else
```

```
        {
            if (BanderaOD == 1) 4
            {
                LabelError.Text = "(*) El AVCC no debe ser menor que el AVSC, Verifique  los valores del OJO Derecho "; 5
                LabelError.Font.Bold = true; 5
                LabelError.ForeColor = System.Drawing.Color.Red; 5
            }
            if (BanderaOD == 2) 6
            {
                LabelError.Text = "(*)El Eje debe tener una amplitud  menor que 180, Verifique los valores del OJO Derecho "; 7
                LabelError.Font.Bold = true; 7
                LabelError.ForeColor = System.Drawing.Color.Red; 7
            }
            if (BanderaOD == 4) 8
            {
                LabelError.Text = "(*)El valor de la Esfera tiene un formato Incorrecto, Verifique los valores del OJO Derecho"; 9
                LabelError.Font.Bold = true; 9
                LabelError.ForeColor = System.Drawing.Color.Red;  9
            }
            if (BanderaOD == 5) 10
            {
                LabelError.Text = "(*)El valor del Cilindro tiene un formato Incorrecto, Verifique los valores del OJO Derecho"; 11
```

```
            LabelError.Font.Bold = true; 11
            LabelError.ForeColor = System.Drawing.Color.Red; 11

        }

        if (BanderaOD == 3) 12
        {
            if (CilindroDerecho == 1) 13
            {
                ConvertirCilindroPos(); 14
                CilindroDerecho = 2; 14
                ExamenOD(); 14
                new EPRefraccionNegocio().Adicionar(SSExamenRefraccionOD); 14
                ButtonGuardar.Enabled = true; 14
                LabelError.Text = "Registro insertado satisfactoriamente"; 14
                LabelError.Font.Bold = true; 14
                LabelError.ForeColor = System.Drawing.Color.Teal; 14
                ButtonGuardar.Enabled = false; 14
                Label3.Text = ""; 14
            }
            if (CilindroDerecho == 0) 15
            {
                LabelError.Text = "¿Esta seguro de registrar un cilindro con valor positivo para el Ojo
Derecho?"; 16
                LabelError.Font.Bold = true; 16
```

```
            LabelError.ForeColor = System.Drawing.Color.Red; 16
            CilindroDerecho = 1; 16
        }
    }
    else if (BanderaOD == 0) 17
    {
        LabelError.Text = "Debe completar los campos señalados (*)"; 18
        LabelError.Font.Bold = true; 18
        LabelError.ForeColor = System.Drawing.Color.Red; 18
    }
  }
} 19
```

El grafo de flujo asociado al código representado anteriormente.

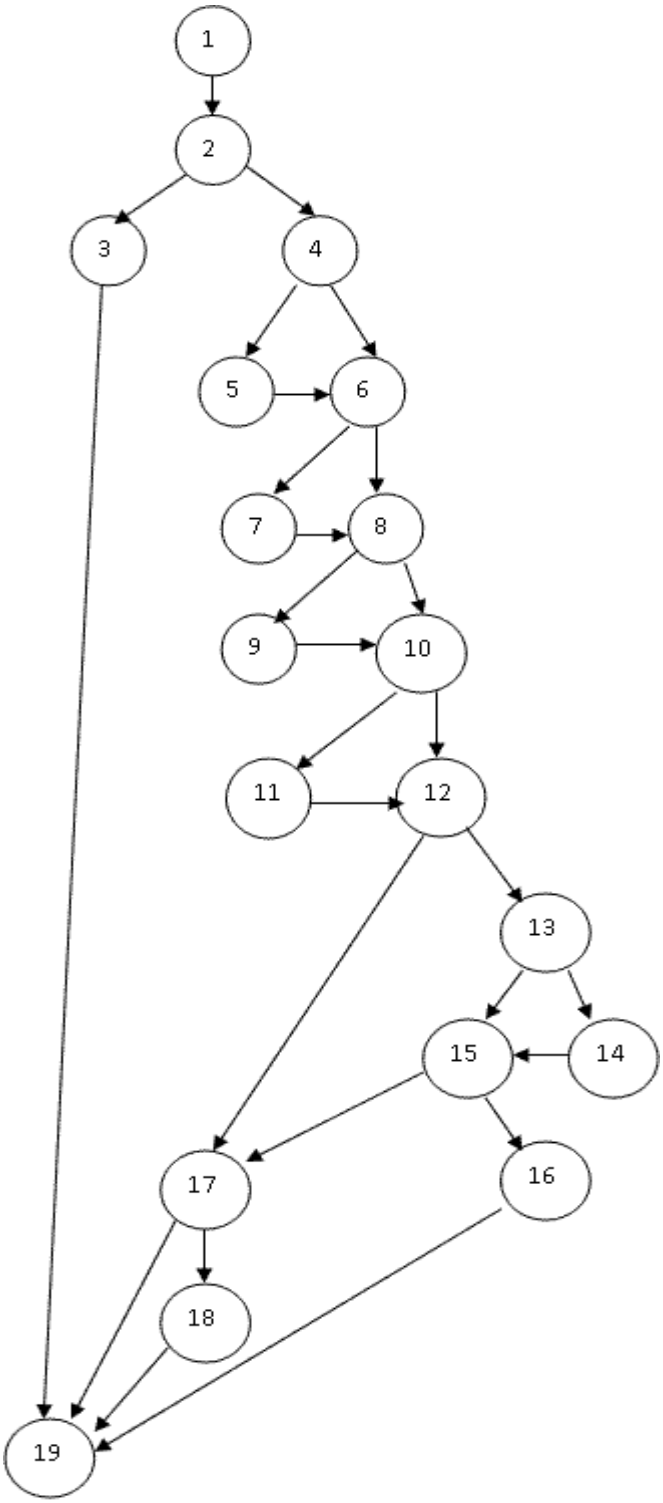

Imagen 2. Grafo de flujo para la función GuardarOjoDerecho.

Se aplica el cálculo mediante las tres fórmulas siguientes, el resultado de las tres debe ser el mismo.

- La complejidad ciclomática, V(G), de un grafo de flujo G se define como:

(G) = Aristas – Nodos + 2

V(G) = 26 – 19 +2 = 10

- La complejidad ciclomática, V(G), de un grafo de flujo G se define como:

V(G) = Nodos Predicado + 1

V(G)= 8 + 1 = 10

- El número de regiones del grafo coincide con la complejidad ciclomática, V(G).

VG = Número de regiones

VG = 10

El valor del cálculo efectuado por las diferentes fórmulas ha sido el mismo, para un valor de complejidad ciclomática del código igual a 10, esto significa que existen 10 caminos por donde el flujo puede circular.

Este algoritmo tiene como características que en todos los caminos tiene que pasar necesariamente por una condición, en el mejor de los casos solo pasa por 1 que sería la entrada de código insertada por el usuario sea la correcta, el peor de los casos tiene que pasar por 9 condiciones que sería cuando la entrada de datos hecha por el usuario es incorrecta en alguna de las condiciones o en varias. El camino básico que será objetivo de prueba es: 1-2-3-19, este camino garantiza que se ejecute solo la condición necesaria, en caso de ocurrir un error perteneciente a las instrucciones internas, se iría por el camino del tratamiento de errores.

3.3.1 Casos de prueba

Para realizar los casos de prueba es necesario cumplir con las siguientes exigencias:

Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo.

Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.

Entrada: Se muestran los parámetros que entran al procedimiento

Resultados Esperados: Se expone lo que se espera que devuelva el procedimiento.

Caso de prueba.

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: Datos que deben ser introducidos.

Datos del Examen Refracción: (fecha, hora, valor de la Agudeza visual sin corrección (Avsc), valor de la esfera, valor del cilindro, valor del eje, valor de la agudeza visual con corrección (Avcc), valor del Add, valor del DP) , estos datos son obligatorios para el id del ojo a insertar.

Condición de ejecución: Todos los datos serán entrados con los siguientes requisitos.

El valor de la esfera permite negativos, el cilindro permite negativos (en caso de ingresar cilindro positivo, realiza la transposición de la manera: la esfera pasa a ser la suma algebraica del cilindro mas la esfera, al cilindro se le cambia el signo, si el eje menor que 90 sumar 90, si es mayor que 90, restarle 90)
El valor de la esfera y el cilindro aumenta o disminuye de 0.25 en 0.25.
El valor del eje tiene que estar entre 0 y 180.
El valor de la Avcc tiene que ser mayor que la Avsc.
El valor de Add debe aumentar de 0.25 en 0.25 siempre positivo (valores desde +0.75 hasta +3.50).

Entrada:

Datos del examen Refracción

Fecha = 10/06/08, hora = 16:45:58, Avsc = 1.11, esfera = 11.25, cilindro = -11.25, eje 120, Avcc = 2.00, Add = 1.25, Dp = 11

Para el caso de prueba descrito el algoritmo recorrió el mejor de los casos, confirmando su buen funcionamiento mostrando los resultados esperados.

Datos del examen Refracción con error en la entrada de datos.

Fecha = 10/06/2008, hora = 16:45:58, Avsc = 1.11, esfera = 11.20, cilindro = 11.11, eje 200, Avcc = 1.00, Add = 1.25, Dp = 11

Para el caso de prueba descrito el algoritmo recorrió el peor de los casos, analizando todos los caminos del tratamiento de errores.

Luego de aplicar el caso de prueba, se pudo comprobar que el flujo de trabajo del procedimiento está correcto ya que cumple con las condiciones necesarias que se habían planteado para este procedimiento.

3.4 Pruebas de caja negra.

Para comprobar la funcionalidad del sistema, al concluir la codificación del mismo, el equipo de programadores, le realizan una serie de pruebas al software, para así garantizar la Calidad del mismo, uno de estos métodos que se utiliza son la Pruebas de Caja Negra, con las cuales se examinan las especificaciones que señalan lo que el programa debe hacer y como lo debe llevar a cabo. La prueba no se hace en base al código, sino a la interfaz. No importa cubrir todas las rutas dentro del programa lo importante es probar todas las entradas en sus valores válidos e inválidos y lograr que el sistema tenga una interfaz amigable.

Se escoge como caso de estudio el caso de uso: “Gestionar exámenes especializados”. El cual consiste en una vez registrado un paciente en el sistema, los especialistas podrán acceder desde la consulta o desde el menú principal a consultar o ingresar exámenes del paciente previamente seleccionado.

Objetivo del test:

Validar el caso de uso Gestionar exámenes especializados.

Las pruebas realizadas de este caso de uso son:

- Insertar nuevo examen especializado.
- Consultar examen especializado.

Flujo Central

El flujo central no es más que la interacción usuario y aplicación, paso a paso, el pedido y la respuesta.

- El especialista accede a la página BuscarPaciente para buscar a un paciente el cual desea atender.
- El sistema muestra los datos a introducir para la búsqueda.
- El especialista inserta los datos necesarios para la búsqueda.

- El sistema muestra el paciente o la lista de pacientes con datos similares en caso de que la búsqueda sea demasiado amplia.
- El especialista selecciona el tipo de examen a efectuar.
- El sistema muestra una lista de los exámenes del tipo de examen seleccionado**,** realizados al paciente.
- El especialista verificará por la fecha si es necesario efectuar un nuevo examen o consultar alguno realizado recientemente.
- El especialista decide consultar un examen y da clic encima del mismo.
- El sistema muestra los datos de dicho examen.
- El especialista después de verificar los exámenes, pulsa el botón “Salir”.

Flujo Alterno

- El especialista decide insertar un nuevo examen e introduce los datos necesarios.
- El especialista pulsa el botón “Guardar examen”.
- El sistema muestra un mensaje “Los datos han sido guardados satisfactoriamente”.

Precondiciones

Tienen que estar presente en la base de datos.

- Registro de pacientes.
- Registro de autorización para el acceso al módulo de exámenes.
- Registro de ID de Ojos.

Sección: Insertar nuevo examen especializado (Refracción VAP)					
Clases válidas	**Clases no válidas**	**Resultados de la prueba.**	**Resultados de la prueba.**	**Observaciones**	**Cumplimiento**

El especialista selecciona la opción “Exámenes especializados” del menú principal de trabajo.		Muestra la interfaz de exámenes especializados	Satisfactoria	La operación se realizo correctamente	100%
El especialista inserta los datos específicos del examen a realizar Agudeza Visual sin corrección (Avsc) = 1.00	.	El sistema registra los datos.	Satisfactoria	La operación se realizó correctamente	La operación se realizó correctamente
Esfera = (+/-)1.25		El sistema registra los datos.	Satisfactoria	La operación se realizó correctamente	100%
Cilindro = (+/-)1.25		El sistema registra los datos.	Satisfactoria	La operación se realizó correctamente	100%
Eje = 120		El sistema registra los datos.	Satisfactoria	La operación se realizó correctamente	100%

Agudeza visual con corrección Avcc = 1.10		El sistema registra los datos.	Satisfactoria	La operación se realizó correctamente	100%
Add = 1.25		El sistema registra los datos.	Satisfactoria	La operación se realizó correctamente	100%
Dp = 12		El sistema registra los datos.	Satisfactoria	La operación se realizó correctamente	100%
	El especialista inserta los datos específicos del examen a realizar Esfera = (+/-)1.20	El sistema muestra un mensaje de error determinando el tipo de fallo.	No satisfactoria		
	Cilindro = (+/-)1.11	El sistema muestra un mensaje de error determinando el tipo de fallo.	No satisfactoria		

	Eje = 200	El sistema muestra un mensaje de error determinando el tipo de fallo.	No satisfactoria		

Tabla 52: "Sección: Insertar nuevo examen especializado"

Sección: Consultar examen especializado					
Clases válidas	**Clases no válidas**	**Resultados de la prueba.**	**Resultados de la prueba.**	**Observaciones**	**Cumplimiento**
El especialista selecciona la opción "Exámenes especializados" del menú principal de trabajo.		Muestra la interfaz de exámenes especializados	Satisfactoria	La operación se realizo correctamente	100%
El especialista verifica el listado de exámenes existentes y selecciona el examen deseado.		El sistema muestra los datos.	Satisfactoria	La operación se realizo correctamente.	100%

Tabla 53: "Sección: Consultar examen especializado"

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En este capítulo se realizó un análisis sobre varios métodos de pruebas, específicamente en los usados en el desarrollo de la aplicación. Las pruebas realizadas fueron las de caja blanca y caja negra, demostrando estas un adecuado funcionamiento de la misma. Estas acciones ejecutadas durante la implementación tienen el objetivo de verificar y asegurar que el sistema implementado sea correcto, seguro y tenga calidad, logrando así una mayor satisfacción en el cliente final.

CONCLUSIONES

Con el desarrollo de esta investigación se arribó a las siguientes conclusiones:

1. Se implementó la capa de acceso a datos de los servicios oftalmológicos Cirugía del Cristalino, Refractiva y Pterigium, así como los módulos de Administración y Configuración del Sistema.
2. Se implementó las funcionalidades de los servicios y módulos antes mencionados, brindando una interfaz gráfica orientada al usuario.
3. Se logró la integración con el componente SAAA (EMP) viabilizando así la seguridad del Sistema.
4. Se llevó a cabo una profunda investigación de las principales tecnologías utilizadas en el desarrollo de aplicaciones web.
5. El estudio de los estándares de codificación fue de gran utilidad en la implementación, sirvió para aprender nuevos métodos y formas de tener organizado el código, facilitando las futuras actualizaciones mantenimientos o iteraciones para mejorar la aplicación.

De esta forma se le da respuesta a todas las tareas trazadas para la elaboración de este trabajo de diploma, obteniendo así la implementación de nuevos servicios para el Bloque Quirúrgico Oftalmológico, elevando así la calidad y seguridad del mismo.

RECOMENDACIONES

Los autores recomiendan:

1. Implantación del sistema en los hospitales oftalmológicos cubanos.
2. Continuar agregando servicios oftalmológicos al sistema, para una mayor usabilidad del mismo.
3. Preparar al cliente para el futuro uso de la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

1. Medicas., R.H.d.C. 2004. p. http://www.ucmh.sld.cu/rhab/editorial_rev10.htm.
2. MINREX. [citado; Disponible en: http://www.cubacoop.com/cubacoop/misionmilagros.htm.
3. Médicos, P. *Programas o Planes de las distintas especialidades médicas, según el modelo del Ministerio de Sanidad y Consumo de España* [citado; Disponible en: http://www.portalesmedicos.com/plan_programa_especialidad/oftalmologia_1.htm.
4. QSOFT. [citado; Disponible en: http://www.oftalmosalus.com/.
5. Medisys. [citado; Disponible en: http://www.oftal.com/.
6. ActualSoft! [citado; Disponible en: http://www.actualsoft.com.ar/plus.htm.
7. VisionDat. [citado; Disponible en: http://www.visiondat.com/index.php?mod=visiondat.
8. Web, A.e.l. 2007 [citado; Disponible en: http://ayudaenlaweb.blogspot.com/2007/10/qu-es-un-navegador-web.html
9. Müller, P. *Una Revisión a las Técnicas de Programación*. 1997 [citado; Disponible en: http://www.gnacademy.org/text/cc/Tutorial/Spanish/node3.html.
10. Feijoo, B.S. [citado; Disponible en: http://teleformacion.edu.aytolacoruna.es/PASCAL/document/modular.htm#intro.
11. *Lenguajes de programación*. [citado; Disponible en: http://www.lenguajes-de-programacion.com/programacion-orientada-a-objetos.shtml.
12. Hinostroza, R.R. *Características de PHP*. 2007 [citado; Disponible en: http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.
13. Salazar, D.J.S. *Laboratory of Computational Genomics*. 2005 [citado; Disponible en: http://tikal.cifn.unam.mx/%7Ejsegura/.
14. Microsoft. *Visual C# 2005*. 2005 [cited; Available from: http://msdn.microsoft.com/es-es/library/kx37x362(VS.80).aspx.
15. *ASP .NET*. [citado; Disponible en: http://www.miliuco.net/aspnet/aspnet_intro.html.
16. Heredia, M.G. [citado; Disponible en: http://www.tufuncion.com/diferentes-lenguajes-programacion.
17. Francisco., M. [citado; Disponible en: http://www.monografias.com/trabajos6/ixml/ixml.shtml#xml.
18. Microsoft. [citado; Disponible en: http://www.microsoft.com/downloads/details.aspx?displaylang=es&FamilyID=0856eacb-4362-4b0d-8edd-aab15c5e04f5.

19. Cruz, A.D.V. [citado; Disponible en: http://www.elguille.info/colabora/NET2005/vecrado_CommonLanguageRuntime.htm.
20. Planet, R.C. [citado; Disponible en: http://www.channelplanet.com/index.php?idcategoria=12724.
21. Microsoft. *Entorno de desarrollo integrado*. [citado; Disponible en: http://msdn2.microsoft.com/es-es/library/ms165088(VS.80).aspx.
22. libres, A. [citado; Disponible en: http://alts.homelinux.net/privapp.php?id=19.
23. Arevalo, L.O., *Implementación del Módulo Bloque Quirúrgico Oftalmológico del Sistema de Información Hospitalaria*. 2007.
24. Microsoft. 2008 [citado; Disponible en: http://msdn.microsoft.com/es-es/library/system.web.ui.webcontrols.objectdatasource(VS.80).aspx.
25. Rizzi, F.M. *COMPLEJIDAD CICLOMÁTICA*. [citado; Disponible en: http://www.itba.edu.ar/capis/rtis/articulosdeloscuadernosetapaprevia/RIZZI-COMPLEJIDAD.pdf.
26. Fernández, G. *Estándar codificación DOTNET*. 2005 [citado; Disponible en: http://www.elguille.info/colabora/NET2005/giovannyfernandez_EstandarCodificacionNET.htm.
27. D'Onofrio, D.L. *Probando software y números de versión*. [citado; Disponible en: http://www.elguille.info/Clipper/probando.htm.

BIBLIOGRAFÍA

1. ActualSoft! [citado; Disponible en: http://www.actualsoft.com.ar/plus.htm.
2. Arevalo, L.O., *Implementación del Módulo Bloque Quirúrgico Oftalmológico del Sistema de Información Hospitalaria*. 2007.
3. *ASP .NET*. [citado; Disponible en: http://www.miliuco.net/aspnet/aspnet_intro.html.
4. Cruz, A.D.V. [citado; Disponible en: http://www.elguille.info/colabora/NET2005/vecrado_CommonLanguageRuntime.htm
5. D'Onofrio, D.L. *Probando software y números de versión*. [citado; Disponible en: http://www.elguille.info/Clipper/probando.htm.
6. Feijoo, B.S. [citado; Disponible en: http://teleformacion.edu.aytolacoruna.es/PASCAL/document/modular.htm#intro.
7. Fernández, G. *Estándar codificación DOTNET*. 2005 [citado; Disponible en: http://www.elguille.info/colabora/NET2005/giovannyfernandez_EstandarCodificacionNET.htm.
8. Francisco., M. [citado; Disponible en: http://www.monografias.com/trabajos6/ixml/ixml.shtml#xml.
9. Heredia, M.G. [citado; Disponible en: http://www.tufuncion.com/diferentes-lenguajes-programacion.
10. Hinostroza, R.R. *Características de PHP*. 2007 [citado; Disponible en: http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.
11. *Lenguajes de programación*. [citado; Disponible en: http://www.lenguajes-de-programacion.com/programacion-orientada-a-objetos.shtml
12. libres, A. [citado; Disponible en: http://alts.homelinux.net/privapp.php?id=19.
13. Medicas., R.H.d.C. 2004. p. http://www.ucmh.sld.cu/rhab/editorial_rev10.htm.
14. Médicos, P. *Programas o Planes de las distintas especialidades médicas, según el modelo del Ministerio de Sanidad y Consumo de España* [citado; Disponible en: http://www.portalesmedicos.com/plan_programa_especialidad/oftalmologia_1.htm.
15. Medisys. [citado; Disponible en: http://www.oftal.com/.
16. Microsoft. [citado; Disponible en: http://www.microsoft.com/downloads/details.aspx?displaylang=es&FamilyID=0856eacb-4362-4b0d-8edd-aab15c5e04f5.
17. Microsoft Corporation. Crear aplicaciones ASP .NET seguras, 2004. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/aplic_sec.asp.

18. Microsoft Corporation. Tutorial de ASP.NET, 2001. Disponible en: http://es.gotdotnet.com/quickstart/aspplus/doc/quickstart.aspx

19. Microsoft. *Entorno de desarrollo integrado*. [citado; Disponible en: http://msdn2.microsoft.com/es-es/library/ms165088(VS.80).aspx.
20. Microsoft. *Visual C# 2005*. 2005 [cited; Available from: http://msdn.microsoft.com/es-es/library/kx37x362(VS.80).aspx.
21. Microsoft. 2008 [citado; Disponible en: http://msdn.microsoft.com/es-es/library/system.web.ui.webcontrols.objectdatasource(VS.80).aspx.
22. MINREX. [citado; Disponible en: http://www.cubacoop.com/cubacoop/misionmilagros.htm.
23. OJEDA, F. C. Visual C# .NET. ANAYA, 2002. 641 p.
24. Müller, P. *Una Revisión a las Técnicas de Programación*. 1997 [citado; Disponible en: http://www.gnacademy.org/text/cc/Tutorial/Spanish/node3.html.
25. Planet, R.C. [citado; Disponible en: http://www.channelplanet.com/index.php?idcategoria=12724.
26. QSOFT. [citado; Disponible en: http://www.oftalmosalus.com/.
27. RECIO, FRANCISCO Y PROVENCIO, DAVID: Arquitectura básica de la plataforma .Net. Descripción del Framework y sus principales componentes: Lenguajes, biblioteca de clases y CLR. En http://www.desarrolloweb.com/articulos/1328.php.
28. 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.
29. Rizzi, F.M. *COMPLEJIDAD CICLOMÁTICA*. [citado; Disponible en: http://www.itba.edu.ar/capis/rtis/articulosdeloscuadernosetapaprevia/RIZZI-COMPLEJIDAD.pdf.
30. Salazar, D.J.S. *Laboratory of Computational Genomics*. 2005 [citado; Disponible en: http://tikal.cifn.unam.mx/%7Ejsegura/.
31. VisionDat. [citado; Disponible en: http://www.visiondat.com/index.php?mod=visiondat.
32. Web, A.e.l. 2007 [citado; Disponible en: http://ayudaenlaweb.blogspot.com/2007/10/qu-es-un-navegador-web.html.
33. http://www.programar.net/. Autor: Ing. Millán Andrés Sánchez Díaz.