Universidad de las Ciencias Informáticas

Facultad 7

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

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Leonexy Oliva Arévalo

Tutores: Lic. Dainerys Castañeiro Rodríguez

Ing. Mauricio G. Espinosa Robaina

Ciudad de La Habana, Julio 2007

“Nunca consideres el estudio como un deber,

sino como una oportunidad para penetrar en el maravilloso mundo del saber.”

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Leonexy Oliva Arévalo Lic. Dainerys Castañeiro Rodríguez

_____

Ing. Mauricio G. Espinosa Robaina

DATOS DE CONTACTO

Tutores:

Lic. Dainerys Castañeiro Rodríguez. Profesora graduada en Licenciatura en Ciencias de la Computación en el año 2004. Ha impartido asignaturas de Sistemas de Bases de Datos, Ingeniería de Software y Gestión de Software. Posee categoría docente de Instructor y cursa la maestría de Ciencias de la Computación.
Email: dainerysc@uci.cu

Ing. Mauricio G. Espinosa Robaina. Graduado de Ingeniero Electrónico, especialidad Máquinas Computadoras en 1985. Administrador de Red en el Instituto Cubano de Oftalmología “Ramón Pando Ferrer”. Jefe de Proyecto para la Confección de Historia Clínica Computarizada de Oftalmología.
Email: mger@infomed.sld.cu

Agradecimientos

Quiero agradecer especialmente:

A mis padres y hermanos, por apoyarme siempre.

A mis compañeros y amigos, que me ayudaron en todos estos años.

A mis familiares, por preocuparse siempre por mí.

A todos los profesores que contribuyeron en mi formación.

A todos los que me ayudaron en la realización de esta investigación.

Al colectivo de trabajo del hospital Pando Ferrer que hizo posible el desarrollo de este trabajo, en especial al Ing. Mauricio G. Espinosa Robaina.

A todos, muchas gracias.

Dedicatoria

A mi mamá, a mi papá y mi hermana, por ser la razón de mí existir.

A Yanet, por su amor y comprensión.

A mi hermano Leonid, por su apoyo.

A Yole por ser primo, amigo, hermano.

Sientan todos que también son autores.

Una parte importante de mis resultados son suyos.

RESUMEN

Actualmente, en los hospitales oftalmológicos cubanos, no existe un sistema informático que gestione la información de los procesos asociados a las intervenciones quirúrgicas de los pacientes, estos se manipulan manualmente, trayendo consigo demora y dificultades para el control de esta información.
Para solucionar este problema, se plantea como objetivo, implementar un sistema automatizado que mejore la gestión de la información referente a los procesos vinculados con las intervenciones quirúrgicas oftalmológicas que tienen lugar en los hospitales del país.

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

En la implementación del sistema, los estándares de codificación fue de gran utilidad, sirvió para aprender nuevos método y formas de tener organizado el código, lo cual ayuda en la comunicación entre desarrolladores, y permite una temprana detección de errores, facilitando las futuras actualizaciones mantenimientos o iteraciones para mejorar la aplicación.

Durante la implementación, se realizaron pruebas, con el objetivo de asegurar que el sistema esté correcto y tenga calidad, para así lograr mayor satisfacción en el cliente final.

Además se espera que el sistema propuesto sea utilizado en los centros hospitalarios del país, optimizando los servicios oftalmológicos y contribuyendo a una mejor organización y menor tiempo de espera por parte de los pacientes.

Palabras Claves: Sistema de Información Hospitalaria, Bloque quirúrgico oftalmológico.

ÍNDICE

INTRODUCCIÓN

A partir de 1996, se dan los primeros pasos para el ordenamiento de un trabajo continuo destinado a impulsar el uso y desarrollo de las Tecnologías de la Información y las Comunicaciones en el país, estos pasos, aunque discretos, condujeron en enero de 2000 a la creación del Ministerio de la Informática y las Comunicaciones con la misión fundamental de fomentar el uso masivo de las Tecnologías de la Información y las Comunicaciones para satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad, tomando como sustento los cuatro pilares fundamentales del proceso revolucionario cubano: *la educación,* ***la salud****, la seguridad social y la cultura*. [1]

Para ellos se cuenta con varias instituciones entre las que se encuentra la Universidad de las Ciencias Informáticas, un centro que vincula la enseñanza a la producción y a solo cinco años de creada, ya tiene sus primeros impactos en la industria cubana del software.

La Universidad está dividida por facultades y estas a su vez por áreas temáticas. La facultad 7 está destinada a la informatización del sector de la salud y se le dio la tarea de realizar un Sistema de Información Hospitalaria, que integre todos los módulos de un hospital. Uno de estos módulos es el Bloque Quirúrgico que se encarga de registrar todos los procesos por los que transcurre un paciente que será intervenido quirúrgicamente.

Este trabajo trata sobre los servicios de cirugía oftalmológica, que tienen gran importancia y han alcanzado mayor auge con el desarrollo de la Misión Milagro que se lleva a cabo tanto dentro como fuera del país. La misma, 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.

Actualmente, el procesamiento de esta información se realiza manualmente, dificultando el trabajo de los médicos, además de que debido a la gran cantidad de datos se pueden cometer errores. Los archivos de información se mantienen creciendo cada día, llegando en algunos casos, a duplicarse. Estos datos en formato duro pueden en ocasiones pueden deteriorarse o en el peor de los casos, extraviarse. La impresión de reportes, informes, documentos que serán archivados en la historia clínica y otros, es muy

lenta al igual que el manejo de estos datos, lo que hace que la espera de los pacientes sea mayor. Debido a esto, los servicios en el hospital en ocasiones no son los esperados. En el Hospital Oftalmológico Ramón Pando Ferrer, se cuenta con una herramienta para gestionar parte de la información, tratando de agilizar estos procesos y optimizarlos, sin embargo no se logra porque esta cumple con funcionalidades estrictas, sin ser propensas al cambio. Está desarrollada en Clipper, basado en MS-DOS, presenta una interfaz gráfica poco amigable, de difícil configuración. Debido a los cambios realizados en los hospitales oftalmológicos, actualmente este sistema resulta primitivo pues no responde a los intereses de los usuarios, por lo que podría declararse obsoleto.

Dada la situación anterior el **problema** radica en ¿Cómo automatizar la gestión de la información relacionada con el proceso quirúrgico oftalmológico que tiene lugar en los hospitales cubanos?

El **objeto de estudio** se centra en la automatización de los procesos vinculados con las intervenciones quirúrgicas oftalmológicas.

El **campo de acción** apunta al proceso de automatización de la información vinculada con las intervenciones quirúrgicas oftalmológicas que tiene lugar en los hospitales cubanos.

Par dar solución al problema antes mencionado se propone como **objetivo general**: Implementar un sistema automatizado que mejore la gestión de la información referente a los procesos vinculados con las intervenciones quirúrgicas oftalmológicas que tienen lugar en los hospitales del país.

Entre los **objetivos específicos** se plantean:

- Analizar los aspectos teóricos conceptuales referente a la implementación del Bloque Quirúrgico Oftalmológico.
- Realizar la implementación de un sistema que controle el proceso de la gestión de la información referente a las intervenciones quirúrgicas oftalmológicas ocurridas en los hospitales.
- Brindar una interfaz gráfica orientada al usuario.
- Garantizar la interoperabilidad y flexibilidad del sistema.

Entre las **tareas de investigación** aparecen**:**

- Hacer un estudio de los sistemas similares existentes tanto nacionales como internacionales.
- Realizar un análisis del Diseño propuesto.
- Hacer un análisis crítico de la tecnología y lenguaje a utilizar.
- Analizar la integración con otros componentes o partes del sistema.
- Realizar la implementación utilizando los patrones de diseño establecidos en el Análisis.
- Realizar las pruebas necesarias para garantizar el correcto funcionamiento de la aplicación implementada.

A continuación se explica la estructuración del contenido de la presente investigación:

El Capítulo I titulado "Fundamentación teórica" ofrece los conceptos básicos asociados al negocio y los métodos científicos que se utilizaron para llevar a cabo el conocimiento del negocio.
Se brinda el estado del arte en cuanto a técnicas, tecnologías, metodologías y software usados 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, un 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

Este capítulo está dedicado a realizar 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 Sistemas internacionales.

En la actualidad existen múltiples sistemas para la gestión oftalmológica entre los que se destacan: ActualOftalmo!, VisionDat, entre otros.

Se trata de un software propietario destinado a los consultorio oftalmológico, desarrollado por un equipo formado por médicos especialistas y profesionales informáticos, sobre Visual Basic 6.0 y con compatibilidad de la bases de datos DAO (MS Access). [2]

Está conformado por:

- Protocolo Gráfico e interactivo de la Consulta Oftalmológica
- Extensa base de datos de Signos y Síntomas Oftalmológicos
- Registro de Antecedentes Familiares y Personales
- 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
- Vademécum Oftalmológico
- Panel de Alarmas Inteligentes y Manuales
- Recetario de Anteojos.
- Material Instructivo para el paciente

VisionDat es un software de tecnología avanzada que ofrece diversas opciones en el manejo de consultorios. Está destinado a los consultorios especializados en oftalmología y optometría. Software propietario que no brinda información técnica en la cual fue desarrollado. [3]

Está conformado por:
Datos de pacientes.
Datos de consulta.
Citas y actividades.
Expedientes.
Gráficas, estadísticas.

1.2 Sistemas nacionales

Actualmente solo existe un software oftalmológico funcionando en el 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 transitan los paciente. Desarrollado en clípper sobre MS-DOS y bases de datos ficheros mdf.

1.3 Tecnologías

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

1.3.1 Arquitectura Cliente – Servidor

Esta arquitectura consiste básicamente en la combinación de sistemas que pueden colaborar entre si para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde está ubicada. Es una arquitectura de procesamientos cooperativos, donde uno de los componentes pide servicios a otro, existiendo una colaboración entre dos o más computadoras conectadas a una red.

IBM define al modelo Cliente/Servidor. *"Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o clientes, resultan en un trabajo realizado por otros computadores llamados servidores". [4]*

1.3.2 Aplicación Web

Una **aplicación web** es un sistema informático que los usuarios utilizan accediendo a un servidor web a través de Internet o de una intranet. Las aplicaciones web son populares debido a la practicidad del navegador web como cliente ligero. La habilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad.

1.3.3 Técnicas de programación.

Las principales técnicas de programación son:

1.3.3.1 Programación no Estructurada.

Este tipo de programación está basada en una secuencia de instrucciones modificando los datos globales en el transcurso de todo el programa. Donde los saltos y el fin de programa no seguían ninguna estructura. Los saltos podían apuntar a cualquier punto del código, lo que ocasionaba que el algoritmo terminara siendo un ovillo indescifrable. Tampoco se podía saber cuándo terminaba.

1.3.3.2 Programación Procedimental.

La programación procedimental es un tipo de programación estructurada en donde el código se divide en porciones llamadas "procedimientos" o "funciones".

1.3.3.3 Programación Modular.

La programación modular está basada en la técnica de diseño descendente, consiste en dividir el problema original en diversos sub-problemas que se pueden resolver por separado, para después recomponer los resultados y obtener la solución al problema.

1.3.3.4 Programación Orientada a Objetos.

La Programación Orientada a Objetos **(POO u OOP** según siglas en inglés) es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan *estado* (datos), *comportamiento* (procedimientos o *métodos*) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulo más fáciles de escribir, mantener y reutilizar. De esta forma, un objeto contiene toda la información, (los denominados atributos) que permite definirlo e identificarlo frente a otros objetos

pertenecientes a otras clases (e incluso entre objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos). A su vez, dispone de mecanismos de interacción (los llamados métodos) que favorecen la comunicación entre objetos (de una misma clase o de distintas), y en consecuencia, el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan (ni deben separarse) información (datos) y procesamiento (métodos). [5]

1.3.4 Lenguajes y plataformas de desarrollo

Un lenguaje de programación es utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.

Un lenguaje de programación permite a un programador especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural, tal como sucede con el lenguaje Léxico.

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

1.3.4.1 Lenguaje del lado del cliente.

Dentro del grupo de lenguajes del lado del cliente algunos de los más usados 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).

1.3.4.2 JavaScript

Javascript es un lenguaje interpretado, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas web.

1.3.4.3 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.

1.3.4.4 AJAX

AJAX (XML y JavaScript asíncronos) no es una tecnología. 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.

1.3.4.5 Lenguaje de Marcas Extensible (XML)

Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del Lenguaje de Marcación Generalizado (SGML) y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

1.3.4.6 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.

1.3.4.7 Lenguaje del lado del servidor.

Dentro del grupo de lenguajes del lado del servidor los más usados a nivel mundial son ASP, ASP.NET, 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).

1.3.4.8 PHP

Es un lenguaje de programación usado generalmente para la creación de contenido para sitios Web. PHP es un acrónimo recurrente que significa "Hypertext Pre-processor" y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web.

1.3.4.9 Active Server Pages (ASP)

Es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS).

1.3.4.10 ASP.NET

Es un conjunto de tecnologías de desarrollo de aplicaciones web comercializado por Microsoft. Es usado por programadores para construir sitios web domésticos, aplicaciones web y servicios XML. Forma parte de la plataforma .NET de Microsoft y es la tecnología sucesora de la tecnología Active Server Pages (ASP). Posee un mejor rendimiento, eficacia y flexibilidad**.** Es un código de Common Language Runtime compilado que se ejecuta en el servidor.
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. [6]

1.3.4.11 PERL

Es un lenguaje de propósito general originalmente desarrollado para la manipulación de texto y que ahora es utilizado para un amplio rango de tareas incluyendo administración de sistemas, desarrollo web, programación en red, desarrollo de interfaces graficas de usuario y más.
Es muy utilizado para construir aplicaciones CGI para la Web. Es software libre, es un lenguaje de programación interpretado y extensible a partir de otros lenguajes.

1.3.5 Plataformas

1.3.5.1 Net Framework

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. [7]

El .NET Framework fue diseñado para cumplir con objetivos como: proporcionar un entorno coherente de programación orientada a objetos, donde 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, la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza, que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos; basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código. .NET tiene dos componentes principales: Common Language Runtime (CLR) y la biblioteca de clases de .NET Framework (BCL - Basic Class Library). **(Ver anexo 1).**

1.3.5.2 El Common Language Runtime (CLR)

El CLR administra la memoria, ejecución de subprocesos, ejecución de código, comprobación de la seguridad del código, compilación y demás servicios del sistema. Es la máquina virtual de .NET. El código destinado al motor de tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado.

1.3.5.3 La Biblioteca de Clases de .NET

La biblioteca de clases de .NET Framework es una colección de tipos reutilizables que se integran estrechamente con el CLR. 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.
Esta librería está escrita en MSIL (Microsoft Intermediate Language – Lenguaje Intermedio de Microsoft) que es un conjunto de instrucciones independiente del CPU que se pueden convertir de forma eficaz en código nativo. Por tanto esta librería puede ser utilizada desde cualquier lenguaje cuyo compilador genere MSIL. **(Ver anexo 2)**

1.3.5.4 Mono

Es una plataforma de desarrollo de código abierto basada en .NET Framework. Es una plataforma para ejecutar y desarrollar aplicaciones modernas basadas en los estándares ECMA/ISO. Puede ejecutar aplicaciones hechas para los Framework .NET y Java.
Permite a los desarrolladores construir aplicaciones multiplataforma de alta productividad. Acoge varias licencias libres (X11, LGPL y GPL) y tiene una comunidad de desarrolladores activa, está patrocinado por Novell y es la base para muchas aplicaciones. Incluye compiladores, un intérprete compatible con el CLR de ECMA y un conjunto de librerías. Las librerías cubren la compatibilidad con Microsoft .NET (incluyendo ADO.NET, System.Windows.Forms y ASP.NET).
Mono posee librerías adicionales y otras de terceras partes. Para el desarrollo de aplicaciones gráficas se incluye la librería GTK# que es un "binding" sobre GTK+ y GNOME para permitir el desarrollo de aplicaciones nativas para Gnome utilizando Mono y cualquiera de los lenguajes soportados. El diseño de interfaces gráficas se puede realizar con Glade. Además está Monodevelop que es un IDE (Entorno de Desarrollo Integrado) que integra la gestión de proyectos, la construcción de aplicaciones, depurador y toda la documentación (APIs, especificaciones, etc.)

1.3.6 Entorno de Desarrollo Integrado

1.3.6.1 Visual Studio 2005

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

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

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

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

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

1.3.6.2 SharpDevelop

Entorno Integrado de Desarrollo (IDE) libre. Es un proyecto que comenzó desde los comienzos del lenguaje C# en el año 2000, soporta C#, Visual Basic .NET, ASP.NET, XML. Es de diseño visual al igual que el Visual Studio, y bastante similar, es compatible con Net Framework 1.1, 2.0, Compact Framework y Mono, viene con depuradores, configuración de proyectos, además de ser escrito enteramente en C#.

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
- Tambien incluye sintaxis coloreada, paréntesis inteligentes, bookmarks, plantillas, herramientas para expresiones regulares, asistentes, exportación HTML, visor de clases, integración con NDoc, integración con Nprof, etc.

En este capítulo se han analizado un grupo de aplicaciones vinculadas con el campo de acción, de la cuales, las internacionales no se adecuan al sistema de salud cubano y al ser propietarias no permiten adaptaciones, además, su despliegue constituiría una gran inversión de recursos. De este tipo de software solo existe uno nacional, que está en funcionamiento en el Hospital Oftalmológico Ramón Pando Ferre. El mismo no satisface las necesidades actuales de las instituciones oftalmológicas, pues solo resuelve una parte del proceso.

Conjuntamente, se han analizado un grupo de tecnologías y lenguajes candidatos, se decidió por 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 la metodología 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.

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

En este capítulo se aborda el análisis del diseño propuesto por los analistas. Se analizaran las posibles implementaciones de módulo y componentes, así como la reutilización de los existentes. Se harán descripciones de las clases, los tipos de datos y las operaciones que se implementen para dar solución al problema.

2.1 Valoración crítica del diseño propuesto por el analista.

Del diseño propuesto por los analista su pudo extraer las clases fundamentales que deben ser definidas para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, dando una idea clara de lo que se debe implementar. Unido a esto se encuentran los diagramas de interacción, que explican la colaboración que existe entre las clases y cómo son llamados los métodos y sentencias dentro de cada una, de los cuales se obtuvo la información necesaria para conocer el orden de las acciones a implementar.

El diseño propuesto fue creado siguiendo patrones, que de manera general constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, permitiendo llevar a cabo la implementación clara y limpia del módulo bajo patrones como los GRASP y los GOF.

Los patrones GRASP se utilizan con el objetivo de asignar responsabilidades a las diferentes clases que se definen en el diseño. Dentro de este grupo se identifican cinco patrones muy utilizados: Experto, Creador, Alta Cohesión, Bajo Acoplamiento y el Controlador. Estos patrones se les aplican a las clases definidas en el diseño, distribuyendo responsabilidades entre las mismas de forma tal que no existan muchas relaciones, que no se sobrecargue de métodos a una clase en específico pudiendo acomodarlos en otras, entre otras mejoras que brinda el uso de este grupo de patrones.[8]

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

la pandilla de los cuatro o simplemente GOF, se propone el uso de los patrones Fabricación Pura y Fábrica Abstracta, este último proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas y el anterior asigna un conjunto altamente cohesivo de responsabilidades a una clase artificial que no representa nada en el dominio del problema, una clase creada para dar soporte a una alta cohesión, un bajo acoplamiento y reutilización.[9]

2.2 Análisis de posibles implementaciones, componentes o módulo ya existentes y 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.

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

Entre estos módulos están:

Configuración: Encargado de gestionar toda la información básica referente al proceso quirúrgico oftalmológico, por ejemplo: los antecedentes patológicos, lugar colocación del cristalino, las distintas patología, entre otros. **(Ver anexo 3)**

Bloque Quirúrgico General: Gestiona toda la información común para todas la especialidades. La consulta especializada está a cargo de los datos generales de la consulta como son: motivo de la consulta, historia de la enfermedad actual, reconsulta, etc. **(Ver anexo 4)**. En el anuncio operatorio gestiona los datos del paciente y el personal de cirugía. **(Ver anexo 5, 6)**

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 paciente, 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. **(Ver anexo 7)**

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

Chameleon v4.1: Responsable de la mensajería HL7.

Está compuesto por un conjunto de herramientas que permiten la administración, integridad y flujo de la información médica entre los distintos Sistemas de Información para la Salud (SIS). Este garantiza un fuerte tipado, con notaciones más familiares al usuario y equivalentes a los tipos encapsulados en su núcleo, el usuario no necesita conocer las especificaciones en detalle de HL7 para utilizar el mismo en sus aplicaciones.

Al contar con sus herramientas se realizará el envío y recepción de datos clínicos, información de pacientes y algunos reportes de laboratorio, entre los distintos SIS que lo utilicen, empleando para ello un

solo formato, el especificado por HL7. Reduce los recursos invertidos en la negociación de las interfaces entre aplicaciones para la salud.

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, en caso de utilizarse servicios web, la información que es transmitida debe cumplir con los estándares internacionales que hay establecidos para facilitar la integración entre nuevos componentes y otros sistemas hospitalarios.

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

2.4 Descripción de los algoritmos no triviales a implementar.

La cirugía del cristalino es uno de los servicios que será automatizado por el Bloque Quirúrgico Oftalmológico. La misma puede ser una cirugía con implante de lento intraocular o no. En la cirugía del Cristalino con implante de lente intraocular, debe conocerse el estado del ojo contrario para tomar decisiones en el cálculo del lente a implantar al paciente. Debe valorarse por parte del oftalmólogo la esfera del otro ojo para de esa forma determinar que refracción debe esperarse para el ojo a operar, estando sujeto al tipo de lente a colocar: Cámara Anterior o Cámara Posterior. La esfera esperada se calcula a través de fórmulas mundialmente reconocidas que brindan un pronóstico de la esfera con que debe quedar el paciente. Los parámetros que incluyen las fórmulas son: Constante A (A), Factor del Cirujano (SF) y Profundidad de la Cámara Anterior (ACD), la Constante A pueden ser tomada de los datos de fabricación del lente o estimados a través de cálculos matemáticos para su personalización.

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. La métrica, propuesta por Thomas McCabe en 1976, 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 mas 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**. [10]

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

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:

```
public List<ResultadoFormula> CalcularRefraccion(EPOtrosExamenesOftalmologicos examen)
{ 1
   List<ResultadoFormula> listResultado = new List<ResultadoFormula>(); 1
   if (examen != null && examen.Biometria != null && examen.Queratometria != null) 2
   {
      ResultadoFormula resultTMP = null;  3
      Double al = examen.Biometria.LongitudAxial.Value;  3
      Double k = (examen.Queratometria.EjeDebil.Value +
                        examen.Queratometria.EjeFuerte.Value) / 2;  3
      foreach (FormulaDioptrica var in listFD)  4
      {
         resultTMP = new ResultadoFormula();  5
         resultTMP.IdFormula = var.IdFormulaDioptria.Value;  5
```

```
        resultTMP.NombreFormula = var.Descripcion;  5
        resultTMP.Resultado = new List<ResultadoCalculo>();  5
        Double refraccion = 0;  5
        for (double i = 0; i <= 40; I += 0.5)  6
        {
            switch (var.Descripcion)  7
            {
                case "Srk_t":  8
                        refraccion = Math.Round(Srk_t(al, k, constante, i), 2);  9
                    break;
                case "Srkt_II":  10
                        refraccion = Math.Round(Srkt_II(al, k, constante, i), 2);  11
                    break;
                case "Holladay":  12
                        refraccion = Math.Round(Holladay(al, k, constante, i), 2);  13
                    break;
                case "Hoffer_q":  14
                        refraccion = Math.Round(Hoffer_q(al, k, constante, i), 2);  15
                    break;
                case "Binkhorst_II":  16
                   refraccion = Math.Round(Binkhorst_II(al, k, constante, i),   2);  17
                    break;
            }
            resultTMP.Resultado.Add(new ResultadoCalculo(refraccion - 2.0, refraccion, 0, 0, i));  18
        }
        listResultado.Add(resultTMP);  19
    }
}
else
{
    throw new Exception("Faltan valores para realizar el cálculo");  20
```

```
    }
    return listResultado;   21
} 22
```

Después de este paso, es necesario representar el grafo de flujo asociado (Imagen 1), en el cual se representan distintos componentes como son los círculos que se denominan 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.

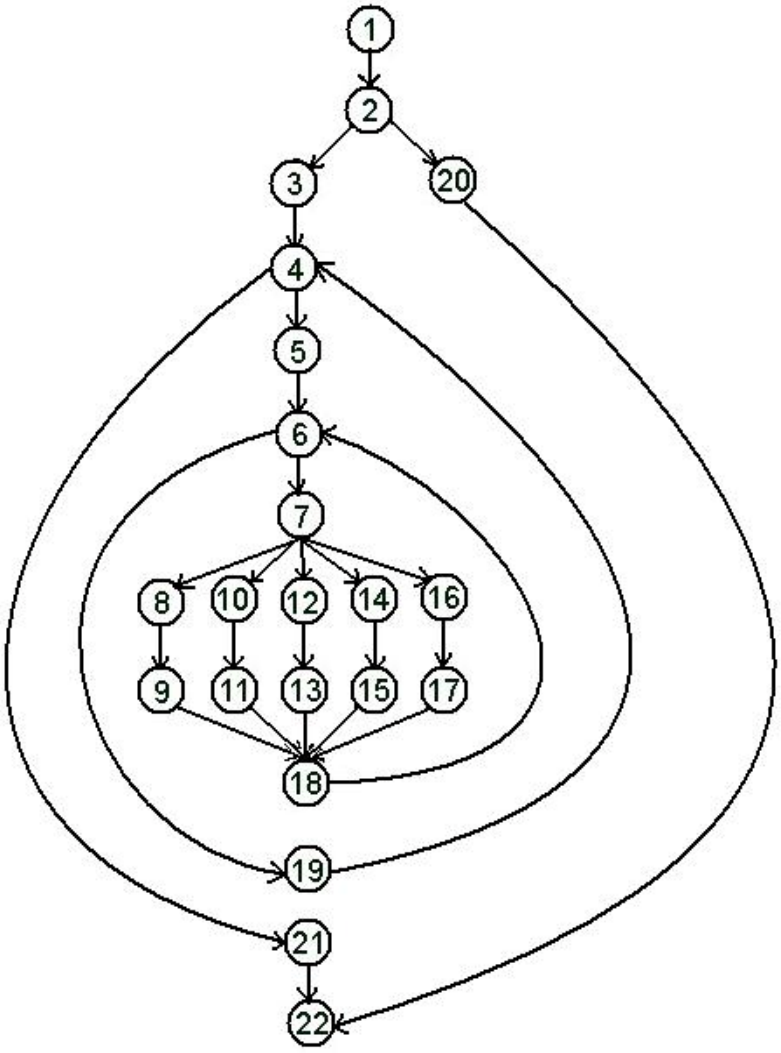


Imagen 1. Grafo de flujo del algoritmo.

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

V (G) = A – N + 2

Donde A es el número de aristas en el grafo, N es el número de nodos. V se refiere al número ciclomático en teoría de grafos y G indica que la complejidad es una función del grafo.

V (G) = (28 – 22) + 2

V (G) = 8

V (G) = P + 1

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

V (G) = 7 +1

V (G) = 8

V (G) = R

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

V (G) = 8

Realizado el cálculo por las tres vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática de 8, dando una visión de que existen a lo sumo ocho caminos lógicos por donde recorrer el algoritmo. Al tener una complejidad ocho, la evaluación de riesgo no dice que es un programa simple, sin mucho riesgo.

2.4.2 Selección de las estructuras de datos apropiadas para la implementación de estos algoritmos. Descripción de las clases que se utilicen para representar computacionalmente dicha estructura.

Para llevar a cabo la implementación del algoritmo para el cálculo de refracción fue necesario implementar varias clase, ResultadoFormula y ResultadoCalculo, la primera contiene el nombre de la fórmula aplicada, el identificador de dicha fórmula y una lista de la segunda, y esta contiene la refracción esperada para la cámara anterior y posterior, la disponibilidad de lente para ambas cámaras y la dioptría del lente.

Para listar los resultados se utilizó la lista genérica de .NET.

La lista genérica es una de las nuevas clases de .NET 2.0 está dentro del namespace System.Collections.Generic. Como su propio nombre indica, permite listar cualquier dato, desde un simple listado de Strings hasta un listado de la clase más compleja que se haya creado. Permite funciones muy útiles para ordenar, buscar un índice, comparar, etc.

La lista es una especie de arreglo que se va redimensionando conforme a las necesidades. Al crear la variable List<T> se inicializa su capacidad, que aumenta conforme la lista va creciendo, pero esto es totalmente transparente. El implementador puede hacer la lista tan grande como necesite, y listarlo fácilmente.

La lista permite dado el valor de una posición obtener el elemento que se encuentra en la misma sin tener que recorrerla innecesariamente, admite insertar un objeto en la posición deseada desplazando los demás a la posición inmediata superior, también permite eliminar, adicionar y muestra mediante la propiedad “Count” la cantidad de elementos que contiene.

2.5 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. [11]

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

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

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

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

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

- Las cláusulas, 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[i].Activa )
        {
                //….
        }
}
```

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: Refraccion	
Tipo de clase: Entidad	
Atributo	**Tipo**
idPersona	Int32
numeroConsulta	Int32
agudezaVisualSinCorreccion	Double
esfera	Double
cilindro	Double
eje	Double
agudezaVisualConCorreccion	Double
Para cada responsabilidad:	
Nombre:	Refraccion
Descripción:	Constructor de la clase.
Nombre:	IdPersona
Descripción:	Propiedad para mostrar y modificar el idPersona
Nombre:	NumeroConsulta
Descripción:	Propiedad para mostrar y modificar el numeroConsulta
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

Tabla 1 Clase entidad "Refraccion"

Nombre: TensionOcular	
Tipo de clase: Entidad	
Atributo	**Tipo**
idPersona	Int32
numeroConsulta	Int32
elastotonometria	Double
radioCurvatura	Double
hipertensionArterial	Boolean
Para cada responsabilidad:	
Nombre:	TensionOcular
Descripción:	Constructor de la clase.
Nombre:	IdPersona
Descripción:	Propiedad para mostrar y modificar el idPersona
Nombre:	NumeroConsulta
Descripción:	Propiedad para mostrar y modificar el numeroConsulta
Nombre:	Elastotonometria
Descripción:	Propiedad para mostrar y modificar la elastotonometria
Nombre:	RadioCurvatura
Descripción:	Propiedad para mostrar y modificar el radioCurvatura
Nombre:	HipertensionArterial
Descripción:	Propiedad para mostrar y modificar elhipertensionArterial

Tabla 2 Clase entidad "TensionOcular"

Nombre: Queratometria	
Tipo de clase: Entidad	
Atributo	**Tipo**
idPersona	Int32
numeroConsulta	Int32
meridianoDebil	Double
meridianoFuerete	Double
ejeFuerte	Double
ejeDebil	Double
Para cada responsabilidad:	
Nombre:	Queratometria
Descripción:	Constructor de la clase.
Nombre:	IdPersona
Descripción:	Propiedad para mostrar y modificar el idPersona
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 meridianoFuerete
Nombre:	EjeFuerte
Descripción:	Propiedad para mostrar y modificar el ejeFuerte
Nombre:	EjeDebil
Descripción:	Propiedad para mostrar y modificar el ejeDebil

Tabla 4 Clase entidad “Queratometria”

Nombre: LenteIntraocular	
Tipo de clase: Entidad	
Atributo	**Tipo**
idLio	Int32
idTipoLente	Int32
poderDioptrico	Double
constanteFabricante	String
modeloLente	String
Para cada responsabilidad:	
Nombre:	LenteIntraocular
Descripción:	Constructor de la clase.
Nombre:	IdLio
Descripción:	Propiedad para mostrar y modificar el idLio
Nombre:	IdTipoLente
Descripción:	Propiedad para mostrar y modificar el idTipoLente
Nombre:	PoderDioptrico
Descripción:	Propiedad para mostrar y modificar el poderDioptrico
Nombre:	ConstanteFabricante
Descripción:	Propiedad para mostrar y modificar la constanteFabricante
Nombre:	ModeloLente
Descripción:	Propiedad para mostrar y modificar el modeloLente

Tabla 5 Clase entidad "LenteIntraocular"

Nombre: LenteInforme	
Tipo de clase: Entidad	
Atributo	**Tipo**
idInformeOperatorio	Int32
idLio	Int32
Para cada responsabilidad:	

Nombre:	LenteInforme
Descripción:	Constructor de la clase.
Nombre:	IdInformeOperatorio
Descripción:	Propiedad para mostrar y modificar el idInformeOperatorio
Nombre:	IdLio
Descripción:	Propiedad para mostrar y modificar el idLio

Tabla 6 Clase entidad "LenteInforme"

Nombre: LenteAnuncio		
Tipo de clase: Entidad		
Atributo		**Tipo**
idAnuncioOperatorio		Int32
idLio		Int32
Para cada responsabilidad:		
Nombre:	LenteAnuncio	
Descripción:	Constructor de la clase.	
Nombre:	idAnuncioOperatorio	
Descripción:	Propiedad para mostrar y modificar el idAnuncioOperatorio	
Nombre:	idLio	
Descripción:	Propiedad para mostrar y modificar el idLio	

Tabla 7 Clase entidad "LenteAnuncio"

Nombre: ExamenPadecimiento		
Tipo de clase: Entidad		
Atributo		**Tipo**
idAnatomiaDelOjo		Int32
idPadecimientoOftalmologico		Int32
Para cada responsabilidad:		
Nombre:	ExamenPadecimiento	

Descripción:	Constructor de la clase.
Nombre:	idAnatomiaDelOjo
Descripción:	Propiedad para mostrar y modificar el idAnatomiaDelOjo
Nombre:	idPadecimientoOftalmologico
Descripción:	Propiedad para mostrar y modificar el idPadecimientoOftalmologico

Tabla 8 Clase entidad "ExamenPadecimiento"

Nombre: Biometria	
Tipo de clase: Entidad	
Atributo	**Tipo**
idPersona	Int32
numeroConsulta	Int32
profundidadCamaraAnterior	Double
grosarioDelCristalino	Double
longitudAccial	Double
Para cada responsabilidad:	
Nombre:	Biometria
Descripción:	Constructor de la clase.
Nombre:	idPersona
Descripción:	Propiedad para mostrar y modificar el idPersona
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 9 Clase entidad "Biometria"

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

Tabla 10 Clase entidad "CirugiasAnterioresConsulta"

Nombre: CirugiasAnterior	
Tipo de clase: Entidad	
Atributo	**Tipo**
idCirugiasAnterior	Int32
idTipoCirugia	Int32
tiempoTranscurrido	DateTime
Para cada responsabilidad:	
Nombre:	CirugiasAnterior
Descripción:	Constructor de la clase.
Nombre:	IdCirugiasAnterior
Descripción:	Propiedad para mostrar y modificar el idCirugiasAnterior
Nombre:	IdTipoCirugia

Descripción:	Propiedad para mostrar y modificar el idTipoCirugia
Nombre:	TiempoTranscurrido
Descripción:	Propiedad para mostrar y modificar el tiempoTranscurrido

Tabla 11 Clase entidad "CirugiasAnterior"

Nombre: ExamenOftalmologicoAnatomia	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenOftalmologicoAnatomia	Int32
idPadecimientoOftalmologico	Int32
numeroConsulta	Int32
idPersona	Int32
idOjos	Int32
Para cada responsabilidad:	
Nombre:	ExamenOftalmologicoAnatomia
Descripción:	Constructor de la clase.
Nombre:	IdExamenOftalmologicoAnatomia
Descripción:	Propiedad para mostrar y modificar el idAnatomiaDelOjo
Nombre:	IdPadecimientoOftalmologico
Descripción:	Propiedad para mostrar y modificar el idPadecimientoOftalmologico
Nombre:	NumeroConsulta
Descripción:	Propiedad para mostrar y modificar el numeroConsulta
Nombre:	IdPersona
Descripción:	Propiedad para mostrar y modificar el idPersona
Nombre:	IdOjos
Descripción:	Propiedad para mostrar y modificar el idOjos

Tabla 12 Clase entidad "ExamenAnatomia"

Nombre: ExamenesOftalmologicos	
Tipo de clase: Entidad	
Atributo	**Tipo**
numeroConsulta	Int32
idPersona	Int32
idOjos	Int32
iolMaster	Boolean
Para cada responsabilidad:	
Nombre:	ExamenesOftalmologicos
Descripción:	Constructor de la clase.
Nombre:	NumeroConsulta
Descripción:	Propiedad para mostrar y modificar el numeroConsulta
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

Tabla 13 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 14 Clase entidad "AntecedentesPatologicosOftalmologicos"

Nombre: AnuncioCirugiaCrist	
Tipo de clase: Entidad	
Atributo	**Tipo**
idAnuncioOperatorio	Int32
idFormulasDioptrias	Int32
idLugarColocacion	Int32
idOjos	Int32
afaquia	Boolean
causasNoImplante	String
refraccionEsperada	Double
Para cada responsabilidad:	
Nombre:	AnuncioCirugiaCrist
Descripción:	Constructor de la clase.
Nombre:	IdAnuncioOperatorio
Descripción:	Propiedad para mostrar y modificar el idAnuncioOperatorio
Nombre:	IdFormulasDioptrias
Descripción:	Propiedad para mostrar y modificar el idFormulasDioptrias
Nombre:	IdLugarColocacion
Descripción:	Propiedad para mostrar y modificar el idLugarColocacion
Nombre:	IdOjos
Descripción:	Propiedad para mostrar y modificar el idOjos

Nombre:	Afaquia
Descripción:	Propiedad para mostrar y modificar la afaquia
Nombre:	CausasNoImplante
Descripción:	Propiedad para mostrar y modificar las causasNoImplante
Nombre:	RefraccionEsperada
Descripción:	Propiedad para mostrar y modificar la refraccionEsperada

Tabla 15 Clase entidad "AnuncioCirugiaCrist"

Nombre: InformeOperatorioCristalino	
Tipo de clase: Entidad	
Atributo	**Tipo**
idInformeOperatorio	Int32
idFormeExtraccion	Int32
idIridectomia	Int32
idLugarColocacion	Int32
refraccionEsperada	Double
ipertensionAretrialSecundaria	Boolean
afaquia	Boolean
especificacionNoLente	String
tiempoFaco	DateTime
idOjos	Int32
Para cada responsabilidad:	
Nombre:	InformeOperatorioCristalino
Descripción:	Constructor de la clase.
Nombre:	IdInformeOperatorio
Descripción:	Propiedad para mostrar y modificar el idInformeOperatorio
Nombre:	IdFormeExtraccion
Descripción:	Propiedad para mostrar y modificar el idFormeExtraccion
Nombre:	IdIridectomia

Descripción:	Propiedad para mostrar y modificar el idIridectomia
Nombre:	IdLugarColocacion
Descripción:	Propiedad para mostrar y modificar el idLugarColocacion
Nombre:	RefraccionEsperada
Descripción:	Propiedad para mostrar y modificar la refraccionEsperada
Nombre:	IpertensionAretrialSecundaria
Descripción:	Propiedad para mostrar y modificar la ipertensionAretrialSecundaria
Nombre:	Afaquia
Descripción:	Propiedad para mostrar y modificar la afaquia
Nombre:	EspecificacionNoLente
Descripción:	Propiedad para mostrar y modificar la especificacionNoLente
Nombre:	TiempoFaco
Descripción:	Propiedad para mostrar y modificar el tiempoFaco
Nombre:	IdOjos
Descripción:	Propiedad para mostrar y modificar el idOjos

Tabla 16 Clase entidad "InformeOperatorioCristalino"

Nombre: AnuncioOftalmologico	
Tipo de clase: Entidad	
Atributo	**Tipo**
idAnuncioOperatorio	Int32
idOjo	Int32
idTipoAnuncioQuirurgico	Int32
refraccionEsperada	Int32
ojoUnico	Int32
lentreOtroOjo	Int32
Para cada responsabilidad:	
Nombre:	AnuncioOftalmologico
Descripción:	Constructor de la clase.

Nombre:	IdAnuncioOperatorio
Descripción:	Propiedad para mostrar y modificar el idAnuncioOperatorio
Nombre:	IdOjo
Descripción:	Propiedad para mostrar y modificar el idOjo
Nombre:	IdTipoAnuncioQuirurgico
Descripción:	Propiedad para mostrar y modificar el idTipoAnuncioQuirurgico
Nombre:	RefraccionEsperada
Descripción:	Propiedad para mostrar y modificar la refraccionEsperada
Nombre:	OjoUnico
Descripción:	Propiedad para mostrar y modificar el ojoUnico
Nombre:	LentreOtroOjo
Descripción:	Propiedad para mostrar y modificar el lentreOtroOjo

Tabla 17 Clase entidad "AnuncioOftalmologico"

Nombre: ComplicacionesInforme	
Tipo de clase: Entidad	
Atributo	**Tipo**
idInformeOperatorio	Int32
idComplicaciones	Int32
Para cada responsabilidad:	
Nombre:	ComplicacionesInforme
Descripción:	Constructor de la clase.
Nombre:	IdInformeOperatorio
Descripción:	Propiedad para mostrar y modificar el idInformeOperatorio
Nombre:	IdComplicaciones
Descripción:	Propiedad para mostrar y modificar el idComplicaciones

Tabla 18 Clase entidad "ComplicacionesInforme"

Nombre: CristalinoExamen	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamen	Int32
idParámetroCristalino	Int32
Para cada responsabilidad:	
Nombre:	CristalinoExamen
Descripción:	Constructor de la clase.
Nombre:	IdExamen
Descripción:	Propiedad para mostrar y modificar el idExamen
Nombre:	IdParámetroCristalino
Descripción:	Propiedad para mostrar y modificar el idParámetroCristalino

Tabla 19 Clase entidad "CristalinoExamen"

Nombre: DatosEntrada	
Tipo de clase: Entidad	
Atributo	**Tipo**
idDatosEntrada	Int32
idDatosParámetros	Int32
idParámetroMedicion	Int32
valor	Double
idInformeOperatorio	Int32
Para cada responsabilidad:	
Nombre:	DatosEntrada
Descripción:	Constructor de la clase.
Nombre:	IdDatosEntrada
Descripción:	Propiedad para mostrar y modificar el idDatosEntrada
Nombre:	IdDatosParámetros
Descripción:	Propiedad para mostrar y modificar el idDatosParámetros
Nombre:	IdParámetroMedicion

Descripción:	Propiedad para mostrar y modificar el idParámetroMedicion
Nombre:	Valor
Descripción:	Propiedad para mostrar y modificar el valor
Nombre:	IdInformeOperatorio
Descripción:	Propiedad para mostrar y modificar el idInformeOperatorio

Tabla 20 Clase entidad "DatosEntrada"

Nombre: DatosEquipo	
Tipo de clase: Entidad	
Atributo	**Tipo**
idDatoEquipo	Int32
marca	String
numeroSerie	String
versionSoftware	Double
idUnidad	Int32
nomograma	String
Para cada responsabilidad:	
Nombre:	DatosEquipo
Descripción:	Constructor de la clase.
Nombre:	IdDatoEquipo
Descripción:	Propiedad para mostrar y modificar el idDatoEquipo
Nombre:	Marca
Descripción:	Propiedad para mostrar y modificar el marca
Nombre:	NumeroSerie
Descripción:	Propiedad para mostrar y modificar el numeroSerie
Nombre:	versionSoftware
Descripción:	Propiedad para mostrar y modificar la versionSoftware
Nombre:	IdUnidad
Descripción:	Propiedad para mostrar y modificar el idUnidad

Nombre:	Nomograma
Descripción:	Propiedad para mostrar y modificar el nomograma

Tabla 21 Clase entidad "DatosEquipo"

Nombre: DatosSistema	
Tipo de clase: Entidad	
Atributo	**Tipo**
ablacionPorCapa	Double
scannerOffset	Int32
laserHvIniciarTratamiento	Int32
laserHvFinalizarTratamiento	Int32
energiaLaserAjustada	Double
tratamientoDesdeUltimoTestfluence	Int32
idInformeOperatorio	Int32
Para cada responsabilidad:	
Nombre:	DatosSistema
Descripción:	Constructor de la clase.
Nombre:	AblacionPorCapa
Descripción:	Propiedad para mostrar y modificar la ablacionPorCapa
Nombre:	ScannerOffset
Descripción:	Propiedad para mostrar y modificar el scannerOffset
Nombre:	LaserHvIniciarTratamiento
Descripción:	Propiedad para mostrar y modificar el laserHvIniciarTratamiento
Nombre:	LaserHvFinalizarTratamiento
Descripción:	Propiedad para mostrar y modificar el laserHvFinalizarTratamiento
Nombre:	EnergiaLaserAjustada
Descripción:	Propiedad para mostrar y modificar el energiaLaserAjustada
Nombre:	TratamientoDesdeUltimoTestfluence
Descripción:	Propiedad para mostrar y modificar el tratamientoDesdeUltimoTestfluence

Nombre:	IdInformeOperatorio
Descripción:	Propiedad para mostrar y modificar el idInformeOperatorio

Tabla 22 Clase entidad "DatosSistema"

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 23 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 24 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 25 Clase entidad "ExamenFondoOjo"

Nombre: ExamenMotilidad	
Tipo de clase: Entidad	
Atributo	**Tipo**
idExamenMotilidad	Int32
idParámetroMotiliadOcular	Int32
Para cada responsabilidad:	
Nombre:	ExamenMotilidad
Descripción:	Constructor de la clase.
Nombre:	IdExamenMotilidad
Descripción:	Propiedad para mostrar y modificar el idExamenMotilidad
Nombre:	IdParámetroMotiliadOcular
Descripción:	Propiedad para mostrar y modificar el idParámetroMotiliadOcular

Tabla 26 Clase entidad "ExamenMotilidad"

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 27 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 28 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 29 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 30 Clase entidad "ExamenVitreo"

Nombre: InformeCirugiaRefractiva	
Tipo de clase: Entidad	
Atributo	**Tipo**
idInformeOperatorio	Int32
idTratamientoRefractivo	Int32
idDatoEquipo	Int32
metodo	String
nombreArchivoEntrada	String
temperatura	Double
humedad	Int32
comentario	String
minimoTiempoTratamiento	Int32
tiempoEntrePulsos	Int32
numeroMensajesEyeTracking	Int32
estadoEyeTracking	String

numeroPulsaciones	Int32
numeroCapas	Int32
numeroErrores	Int32
tratamientoRealizadoPorciento	Int32
tratamientoRealizadoValor	Double
tratamientoRealizado	String
idFuncionarioUsuario	Int32

Para cada responsabilidad:	
Nombre:	InformeCirugiaRefractiva
Descripción:	Constructor de la clase.
Nombre:	IdInformeOperatorio
Descripción:	Propiedad para mostrar y modificar el idInformeOperatorio
Nombre:	IdTratamientoRefractivo
Descripción:	Propiedad para mostrar y modificar el idTratamientoRefractivo
Nombre:	IdDatoEquipo
Descripción:	Propiedad para mostrar y modificar el idDatoEquipo
Nombre:	Metodo
Descripción:	Propiedad para mostrar y modificar el metodo
Nombre:	NombreArchivoEntrada
Descripción:	Propiedad para mostrar y modificar el nombreArchivoEntrada
Nombre:	Temperatura
Descripción:	Propiedad para mostrar y modificar la temperatura
Nombre:	Humedad
Descripción:	Propiedad para mostrar y modificar la humedad
Nombre:	Comentario
Descripción:	Propiedad para mostrar y modificar el comentario
Nombre:	MinimoTiempoTratamiento
Descripción:	Propiedad para mostrar y modificar el minimoTiempoTratamiento
Nombre:	TiempoEntrePulsos
Descripción:	Propiedad para mostrar y modificar el tiempoEntrePulsos

Nombre:	NumeroMensajesEyeTracking
Descripción:	Propiedad para mostrar y modificar el numeroMensajesEyeTracking
Nombre:	EstadoEyeTracking
Descripción:	Propiedad para mostrar y modificar el estadoEyeTracking
Nombre:	NumeroPulsaciones
Descripción:	Propiedad para mostrar y modificar el numeroPulsaciones
Nombre:	NumeroCapas
Descripción:	Propiedad para mostrar y modificar el numeroCapas
Nombre:	NumeroErrores
Descripción:	Propiedad para mostrar y modificar el numeroErrores
Nombre:	TratamientoRealizadoPorciento
Descripción:	Propiedad para mostrar y modificar el tratamientoRealizadoPorciento
Nombre:	TratamientoRealizadoValor
Descripción:	Propiedad para mostrar y modificar el tratamientoRealizadoValor
Nombre:	TratamientoRealizado
Descripción:	Propiedad para mostrar y modificar el tratamientoRealizado
Nombre:	IdFuncionarioUsuario
Descripción:	Propiedad para mostrar y modificar el idFuncionarioUsuario

Tabla 31 Clase entidad "InformeCirugiaRefractiva"

Nombre: InformeOperatorioOftalmologico	
Tipo de clase: Entidad	
Atributo	**Tipo**
idInformeOperatorio	Int32
idTipoInformeOftalmologico	Int32
idOjo	Int32
refraccion	Double
Para cada responsabilidad:	
Nombre:	InformeOperatorioOftalmologico

Descripción:	Constructor de la clase.
Nombre:	IdInformeOperatorio
Descripción:	Propiedad para mostrar y modificar el idInformeOperatorio
Nombre:	IdTipoInformeOftalmologico
Descripción:	Propiedad para mostrar y modificar el idTipoInformeOftalmologico
Nombre:	IdOjo
Descripción:	Propiedad para mostrar y modificar el idOjo
Nombre:	Refraccion
Descripción:	Propiedad para mostrar y modificar la refraccion

Tabla 32 Clase entidad "InformeOperatorioOftalmologico"

Nombre: ResultadoCalculo	
Tipo de clase: Entidad	
Atributo	**Tipo**
refCA	Double
refCP	Double
dispCA	Int32
dispCP	Int32
dioptria	Double
Para cada responsabilidad:	
Nombre:	ResultadoCalculo
Descripción:	Constructor de la clase.
Nombre:	RefCA
Descripción:	Propiedad para mostrar y modificar la refCA
Nombre:	RefCP
Descripción:	Propiedad para mostrar y modificar la refCP
Nombre:	DispCA
Descripción:	Propiedad para mostrar y modificar la dispCA
Nombre:	DispCP

Descripción:	Propiedad para mostrar y modificar la dispCP
Nombre:	Dioptria
Descripción:	Propiedad para mostrar y modificar la dioptria

Tabla 33 Clase entidad "ResultadoCalculo"

Nombre: ResultadoFormula	
Tipo de clase: Entidad	
Atributo	**Tipo**
nombreFormula	String
idFormula	Int32
Para cada responsabilidad:	
Nombre:	
Descripción:	Constructor de la clase.
Nombre:	NombreFormula
Descripción:	Propiedad para mostrar y modificar el nombreFormula
Nombre:	IdFormula
Descripción:	Propiedad para mostrar y modificar el idFormula

Tabla 34 Clase entidad "ResultadoFormula"

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. **(Ver anexo 8)**

Nombre: EPAnuncioOperatorioOftalmologico	
Tipo de clase: Entidad	
Atributo	**Tipo**
anuncioOftalmologico	AnuncioOftalmologico
anuncioCirugiaCrist	AnuncioCirugiaCrist
listLenteAnuncio	List<LenteAnuncio>
Para cada responsabilidad:	
Nombre:	AnuncioOperatorioOftalmologico
Descripción:	Constructor de la clase.
Nombre:	AnuncioOftalmologico
Descripción:	Propiedad para mostrar y modificar el anuncioOftalmologico
Nombre:	AnuncioCirugiaCrist
Descripción:	Propiedad para mostrar y modificar el anuncioCirugiaCrist
Nombre:	LenteAnuncio
Descripción:	Propiedad para mostrar y modificar la listLenteAnuncio

Tabla 35 Clase entidad "EPAnuncioOperatorioOftalmologico"

Nombre: EPConsultaEspecializadaOftalmologica	
Tipo de clase: Entidad	
Atributo	**Tipo**
listExamenOftalmologicoEstructuraOjo	List<EPExamenOftalmologicoEstructuraOjo>
listOtrosExamenesOftalmologicos	List<EPOtrosExamenesOftalmologicos>
listAntecedentesPatologicosOftalmologicos	List<AntecedentePatologicoOftalmologico>
listCirugiasAnterioresConsulta	List<CirugiaAnteriorConsulta>
Para cada responsabilidad:	

Nombre:	EPConsultaEspecializadaOftalmologica
Descripción:	Constructor de la clase.
Nombre:	ExamenOftalmologicoEstructuraOjo
Descripción:	Propiedad para mostrar y modificar la list ExamenOftalmologicoEstructuraOjo
Nombre:	OtrosExamenesOftalmologicos
Descripción:	Propiedad para mostrar y modificar la listOtrosExamenesOftalmologicos
Nombre:	AntecedentesPatologicosOftalmologicos
Descripción:	Propiedad para mostrar y modificar la listAntecedentesPatologicosOftalmologicos
Nombre:	CirugiasAnterioresConsulta
Descripción:	Propiedad para mostrar y modificar la listCirugiasAnterioresConsulta

Tabla 36 Clase entidad "EPConsultaEspecializadaOftalmologica"

Nombre: EPExamenCristalino	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenCristalino	ExamenCristalino
listCristalinoExamen	List<CristalinoExamen>
Para cada responsabilidad:	
Nombre:	EPExamenCristalino
Descripción:	Constructor de la clase.
Nombre:	ExamenCristalino
Descripción:	Propiedad para mostrar y modificarel examenCristalino
Nombre:	CristalinoExamen
Descripción:	Propiedad para mostrar y modificar la listCristalinoExamen

Tabla 37 Clase entidad "EPConsultaEspecializadaOftalmologica"

Nombre: EPExamenMotilidadOcular	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenMotilidadOcular	ExamenMotilidadOcular
listExamenMotilidad	List<ExamenMotilidad>
Para cada responsabilidad:	
Nombre:	EPExamenMotilidadOcular
Descripción:	Constructor de la clase.
Nombre:	ExamenMotilidadOcular
Descripción:	Propiedad para mostrar y modificar el examenMotilidadOcular
Nombre:	ExamenMotilidad
Descripción:	Propiedad para mostrar y modificar el listExamenMotilidad

Tabla 38 Clase entidad "EPExamenMotilidadOcular"

Nombre: EPExamenOftalmologicoEstructuraOjo	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenOftalmologicoAnatomia	ExamenOftalmologicoAnatomia
listExamenesAnexos	List<ExamenesAnexos>
listExamenReflejoPupilar	List<ExamenReflejoPupilar>
listExamenVitreo	List<ExamenVitreo>
listExamenSegmentoAnterior	List<ExamenSegmentoAnterior>
listExamenFondoOjo	List<ExamenFondoOjo>
examenMotilidadOcular	EPExamenMotilidadOcular
examenCristalino	EPExamenCristalino
Para cada responsabilidad:	
Nombre:	EPExamenOftalmologicoEstructuraOjo
Descripción:	Constructor de la clase.
Nombre:	ExamenOftalmologicoAnatomia

Descripción:	Propiedad para mostrar y modificar el examenOftalmologicoAnatomia
Nombre:	ExamenesAnexos
Descripción:	Propiedad para mostrar y modificar el listExamenesAnexos
Nombre:	ExamenReflejoPupilar
Descripción:	Propiedad para mostrar y modificar el listExamenReflejoPupilar
Nombre:	ExamenVitreo
Descripción:	Propiedad para mostrar y modificar el listExamenVitreo
Nombre:	ExamenSegmentoAnterior
Descripción:	Propiedad para mostrar y modificar el listExamenSegmentoAnterior
Nombre:	ExamenFondoOjo
Descripción:	Propiedad para mostrar y modificar el listExamenFondoOjo
Nombre:	ExamenMotilidadOcular
Descripción:	Propiedad para mostrar y modificar el examenMotilidadOcular
Nombre:	ExamenCristalino
Descripción:	Propiedad para mostrar y modificar el examenCristalino

Tabla 39 Clase entidad "EPExamenOftalmologicoEstructuraOjo"

Nombre: EPInformeCirugiaCristalino	
Tipo de clase: Entidad	
Atributo	**Tipo**
informeOperatorioCristalino	InformeOperatorioCristalino
lenteInforme	List<LenteInforme>
Para cada responsabilidad:	
Nombre:	EPInformeCirugiaCristalino
Descripción:	Constructor de la clase.
Nombre:	InformeOperatorioCristalino
Descripción:	Propiedad para mostrar y modificar el informeOperatorioCristalino
Nombre:	LenteInforme
Descripción:	Propiedad para mostrar y modificar el lenteInforme

Tabla 40 Clase entidad "EPInformeCirugiaCristalino"

Nombre: EPInformeOperatorioOftalmologico	
Tipo de clase: Entidad	
Atributo	**Tipo**
informeOperatorioOftalmologico	InformeOperatorioOftalmologico
epInformeCirugiaCristalino	EPInformeCirugiaCristalino
informeCirugiaRefractiva	InformeCirugiaRefractiva
Para cada responsabilidad:	
Nombre:	EPInformeOperatorioOftalmologico
Descripción:	Constructor de la clase.
Nombre:	InformeOperatorioOftalmologico
Descripción:	Propiedad para mostrar y modificar el informeOperatorioOftalmologico
Nombre:	InformeCirugiaCristalino
Descripción:	Propiedad para mostrar y modificar el epInformeCirugiaCristalino
Nombre:	InformeCirugiaRefractiva
Descripción:	Propiedad para mostrar y modificar el informeCirugiaRefractiva

Tabla 41 Clase entidad "EPInformeOperatorioOftalmologico"

Nombre: EPOtrosExamenesOftalmologicos	
Tipo de clase: Entidad	
Atributo	**Tipo**
examenesOftalmologicos	ExamenesOftalmologicos
refraccion	Refraccion
biometria	Biometria
queratometria	Queratometria
tensionOcular	TensionOcular
Para cada responsabilidad:	
Nombre:	EPOtrosExamenesOftalmologicos
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
Nombre:	Biometria
Descripción:	Propiedad para mostrar y modificar la biometria
Nombre:	Queratometria
Descripción:	Propiedad para mostrar y modificar la queratometria
Nombre:	TensionOcular
Descripción:	Propiedad para mostrar y modificar la tensionOcular

Tabla 42 Clase entidad "EPOtrosExamenesOftalmologicos"

2.6.2 Clases Controladoras.

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

Tabla 43 Clase controladora “ConsultaEspecializadaOftalmologicaRepositorio”

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

Tabla 44 Clase "ConsultaEspecializadaOftalmologicaSelectionFactory"

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

Tabla 45 Clase "ConsultaEspecializadaOftalmologicaInsertFactory"

Nombre: ConsultaEspecializadaOftalmologicaUpdateFactory	
Tipo de clase:	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	ConsultaEspecializadaOftalmologicaUpdateFactory
Descripción:	Constructor de la clase.
Nombre:	Execute

Descripción:	Actualiza en la BD el objeto de tipo ConsultaEspecializadaOftalmologica pasado por parámetro, requiere del helper.

Tabla 46 Clase "ConsultaEspecializadaOftalmologicaUpdateFactory"

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

Tabla 47 Clase "ConsultaEspecializadaOftalmologicaDeleteFactory"

Nombre: ConsultaEspecializadaOftalmologicaNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
consultaEspecializadaOftalmologicaRepositorio	EPConsultaEspecializadaOftalmologicaRepositorio
anuncioOperatorioftalmologicoRepositorio	EPAnuncioOperatorioOftalmologicoRepositorio
antecedentePatologicoOftalmologicoRepositorio	AntecedentePatologicoOftalmologicoRepositorio
Para cada responsabilidad:	
Nombre:	ConsultaEspecializadaOftalmologicaNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EPConsultaEspecializadaOftalmologica consultaEspecializadaOftalmologica*)
Descripción:	Método que devuelve la consulta especializada oftalmológica correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EPConsultaEspecializadaOftalmologica consultaEspecializadaOftalmologica*)
Descripción:	Método que devuelve una lista de consultas especializadas oftalmológicas

	correspondiante a los datos entrados.
Nombre:	Adicionar(*EPConsultaEspecializadaOftalmologica consultaEspecializadaOftalmologica*)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EPConsultaEspecializadaOftalmologica consultaEspecializadaOftalmologica*)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EPConsultaEspecializadaOftalmologica consultaEspecializadaOftalmologica*)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.
Nombre:	ObtenerUno(*EPAnuncioOperatorioOftalmologico anuncioOperatorioOftalmologico*)
Descripción:	Método que devuelve la anuncio operatorio oftalmológico correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EPAnuncioOperatorioOftalmologico anuncioOperatorioOftalmologico*)
Descripción:	Método que devuelve una lista de anuncios operatorios oftalmológicos correspondiante a los datos entrados.
Nombre:	Adicionar(*EPAnuncioOperatorioOftalmologico anuncioOperatorioOftalmologico*)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EPAnuncioOperatorioOftalmologico anuncioOperatorioOftalmologico*)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EPAnuncioOperatorioOftalmologico anuncioOperatorioOftalmologico*)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 48 Clase controladora "ConsultaEspecializadaOftalmologicaNegocio"

Nombre: AnuncioOperatorioOftalmologicoNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
anuncioOperatorioftalmologicoRepositorio	EPAnuncioOperatorioOftalmologicoRepositorio
Para cada responsabilidad:	
Nombre:	ConsultaEspecializadaOftalmologicaNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EPAnuncioOperatorioOftalmologico anuncioOperatorioOftalmologico*)

Descripción:	Método que devuelve el anuncio operatorio oftalmológico correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EPAnuncioOperatorioOftalmologico anuncioOperatorioOftalmologico*)
Descripción:	Método que devuelve una lista de anuncios operatorios oftalmológicos correspondiante a los datos entrados.
Nombre:	Adicionar(*EPAnuncioOperatorioOftalmologico anuncioOperatorioOftalmologico*)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EPAnuncioOperatorioOftalmologico anuncioOperatorioOftalmologico*)
Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EPAnuncioOperatorioOftalmologico anuncioOperatorioOftalmologico*)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 49 Clase controladora "AnuncioOperatorioOftalmologicoNegocio"

Nombre: InformeOperatorioOftalmologicoNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
epInformeOperatorioOftalmologicoRepositorio	EPInformeOperatorioOftalmologicoRepositorio
Para cada responsabilidad:	
Nombre:	ConsultaEspecializadaOftalmologicaNegocio
Descripción:	Constructor de la clase.
Nombre:	ObtenerUno(*EPInformeOperatorioOftalmologico epInformeOperatorioOftalmologico*)
Descripción:	Método que devuelve el informe operatorio oftalmológico correspondiante a los datos entrados.
Nombre:	ObtenerTodos(*EPInformeOperatorioOftalmologico epInformeOperatorioOftalmologico*)
Descripción:	Método que devuelve una lista de informes operatorios oftalmológicos correspondiante a los datos entrados.
Nombre:	Adicionar(*EPInformeOperatorioOftalmologico epInformeOperatorioOftalmologico*)
Descripción:	Método que inserta en la BD la entidad pasada por parámetro.
Nombre:	Actualizar(*EPInformeOperatorioOftalmologico epInformeOperatorioOftalmologico*)

Descripción:	Método que actualiza en la BD la entidad pasada por parámetro.
Nombre:	Eliminar(*EPInformeOperatorioOftalmologico epInformeOperatorioOftalmologico*)
Descripción:	Método que suprime en la BD la entidad pasada por parámetro.

Tabla 50 Clase controladora "InformeOperatorioOftalmologicoNegocio"

Nombre: CalculoRefraccionFormulasNegocio	
Tipo de clase: Controladora	
Atributo	**Tipo**
constante	Double
listFD	List<FormulaDioptrica>
Para cada responsabilidad:	
Nombre:	CalculoRefraccionFormulasNegocio
Descripción:	Constructor de la clase.
Nombre:	CalcularRefraccion(*EPOtrosExamenesOftalmologicos examen*)
Descripción:	Método que devuelve una lista de resultados, correspondientes al cálculo por las distintas fórmulas, empleando los exámenes entrados por paramentros.
Nombre:	Srkt_II
Descripción:	Fórmula para el calculo del Lente.
Nombre:	Srk_t
Descripción:	Fórmula para el calculo del Lente.
Nombre:	Binkhorst_II
Descripción:	Fórmula para el calculo del Lente.
Nombre:	Holladay
Descripción:	Fórmula para el calculo del Lente.
Nombre:	Hoffer_q
Descripción:	Fórmula para el calculo del Lente.
Nombre:	ACD
Descripción:	Fórmula para el calculo del ADC

Tabla 51 Clase controladora "CalculoRefraccionFormulasNegocio"

2.6.3 Clases interfaces.

Nombre: IUConsultaOftalmologia	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSListConsulta	List<EPConsulta>
SSConsulta	EPConsultaEspecializadaOftalmologica
ControGuardar	Int32
ModificarConsulta	Boolean
PuedeCrearAnuncio	Boolean
SSDatosPersonas	DatosPersonas
SSEPAnuncioOperatorioGeneral	EPAnuncioOperatorioGeneral
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:	ButtonCabecera_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 espesifica 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 espesifica como debe hacer el paginado.
Nombre:	GridViewConsultas_SelectedIndexChanging
Descripción:	Evento que se ejecuta al cambiar la selección del gridview. En el se le espesifica 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 espesifica como debe crear esta fila.

Tabla 51 Clase interfaz "IUConsultaOftalmologia"

Nombre: IUAnuncioOperatorioOftalmologico	
Tipo de clase: Interfaz	
Atributo	**Tipo**
SSDatosPersonas	DatosPersonas
SSConsulta	EPConsultaEspecializadaOftalmologica
SSListAnuncioHistorial	List<EPAnuncioOperatorioGeneral>
VSLugarColocacionCristalino	List<LugarColocacionCristalino>
SSLente	Lente
SSLenteOK	Boolean
SSAnuncioOperatorioOftalmologico	EPAnuncioOperatorioOftalmologico
VSFormulaDioptrica	List<FormulaDioptrica>
PuedeCrearAnuncio	Boolean
SSRefraccion	List<ResultadoFormula>
SSOjoOperar	Ojo
VSSelectedIndex	int
VSPageIndex	int
SSNuevoAnuncio	bool
SSEPAnuncioOperatorioGeneral	EPAnuncioOperatorioGeneral
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:	SelecionarAnuncio

Descripción:	Método para seleccionar uno de los anuncios que hay en el historial, se ejecuta en caso que se acceda a la página a través de la búsqueda de anuncio o a partir de la consulta.
Nombre:	BuscarConsulta
Descripción:	Método para buscar la consulta a la que pertenece el anuncio que activo.
Nombre:	RefrescarHistorialAnuncio
Descripción:	Método para mostrar nuevamente la lista de anuncios del historial.
Nombre:	NuevoAnuncio
Descripción:	Método para crear un nuevo anuncio.
Nombre:	GuardarDatos
Descripción:	Método para guardar toda la información recogida durante la consulta en una variable de sesión.
Nombre:	LimpiarAnuncio
Descripción:	Método para eliminar todos los registros referentes al anuncio operatorio. Se ejecuta cuando se selecciona otro anuncio.
Nombre:	GridViewAnuncio_RowCreated
Descripción:	Evento que se ejecuta cuando el GridView crea cada una de las filas. En el se le espesifica como debe crear esta fila.
Nombre:	GridViewAnuncio_SelectedIndexChanging
Descripción:	Evento que se ejecuta al cambiar la selección del GridView. En el se le espesifica como debe hacer la selección del nuevo elemento.
Nombre:	ButtonCabecera_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: Crear un nuevo anuncio. Adicionar el anuncio actual a la BD Actualizar el anuncio actual en la BD. Ver la consulta a la cual pertenece el anuncio actual.
Nombre:	ButtonTab_Click
Descripción:	Evento que se ejecuta sobre los botones del tab. En este evento se activa el tab correspondiente al boton pulsado

Nombre:	GuardarLente
Descripción:	Método para guardar en sesión la información correspondiente al lente seleccionado.
Nombre:	MostrarLente
Descripción:	Método para mostrar la información del lente correspondiente al anuncio actual.
Nombre:	MostrarFormula
Descripción:	Método para mostrar el nombre de la fórmula dado el identificador.
Nombre:	MostrarIdFormula
Descripción:	Método para mostrar el identificador de la fórmula seleccionada.
Nombre:	ImageButtonBuscarRefraccion_Click
Descripción:	Evento que se ejecuta al hacer click sobre el boton buscar refracción. Mustra el tab correspondiente al calculo de la refracción.
Nombre:	ViewCalculo_PreRender
Descripción:	Evento que se ejecuta antes de ser mostrado el panel correspondiente al cálculo de refracción. En el se obtiene el resultado de este calculo que sera mostrado por las diferentes session para ello.
Nombre:	ButtonFormulas_Click
Descripción:	Evento que se ejecuta al hacer click sobre los botones de los Tab correspondiantes a las fórmulas. Mustra el tab correspondiente a la fórmula seleccionada.
Nombre:	GridView_PreRender
Descripción:	Evento que se ejecuta antes de ser mostrado GridView de los resultados del calculo. En el se le asigna el nombre de la fórmula por la cual se realizaron los cálculos.
Nombre:	GridView_RowCreated
Descripción:	Evento que se ejecuta cuando el GridView crea cada una de las filas. En el se le espesifica como debe crear esta fila.
Nombre:	GridView_SelectedIndexChanging
Descripción:	Evento que se ejecuta al cambiar la selección del GridView. En el se le espesifica como debe hacer la selección del nuevo elemento.
Nombre:	GridView_PageIndexChanging
Descripción:	Evento que se ejecuta al hacer click en el paginado del gridview. En el se le espesifica como debe hacer el paginado.

Nombre:	LinkButtonResultado_Command
Descripción:	Evento que se ejecuta al hacer click sobre los resultados mostrados en el GridView. En el se recoge la informacion del lente correspondiente al resultrado seleccionado.

Tabla 52 Clase interfaz "IUAnuncioOperatorioOftalmologico"

Este capítulo es de gran importancia, en el se describen como está implementado el sistema, así como los módulo que necesita para funcionar. La información brindada en la descripción de las clases que conforman la solución, permite conocer la distribución de las clases, facilitando su entendimiento para futuras actualizaciones.

La descripción del algoritmo para resolver el problema del cálculo de la refracción esperada tiene gran importancia para futuras versiones en aras de mejorar la solución actual. En ella se explica bien detalla la estructura de datos usada, así como la complejidad del algoritmo.

El estudio de los estándares de codificación fue de gran utilidad en la implementación, sirvió para aprender nuevos método y formas de tener organizado el código, lo cual ayuda en la comunicación entre desarrolladores, y permite una temprana detección de errores, facilitando las futuras actualizaciones mantenimientos o iteraciones para mejorar la aplicación.

CAPÍTULO 3: VALIDACIÓN

Uno de los mayores problemas que se afronta actualmente en la esfera de la informática es la calidad del software. El proceso de pruebas es sin dudas unos de los aspectos fundamentales para medir el estado de calidad de un sistema informático.

"En el desarrollo de software, las personas involucradas, cometen errores, suelen equivocarse en algunas formas características, y según algunos autores, hay cierta tasa de errores que es estadísticamente predecible". [12].

3.1 Pruebas a realizar en tiempo de desarrollo.

Pruebas informales o fase de prueba informal, son aquellas pruebas que hace el desarrollador en su puesto de trabajo, tienen como objetivo comprobar que el programa compile y ver que todo está yendo como debiera, normalmente se realizan varios cientos de estas pruebas que básicamente consisten en compilar periódicamente durante el desarrollo y ejecutar para ver el resultado.

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 mas pequeñas, este tipo de pruebas es la mas común.

Las pruebas de integración tienen por objetivo verificar el conjunto funcionamiento de dos o mas módulos, esta se debe poner en práctica desde la creación de dos o mas que interactúen entre si, en el supuesto caso que se necesiten mas de dos 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*).

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, que se trata de encontrar fallas en las que no se atiene a su especificación, como ser interfaz con el usuario, apariencia de los menús, control de las teclas, etcétera.
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 todos 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 las 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.
Para una entrada que está comprendida dentro de un rango, existen tres clases, por debajo, dentro, y por encima del rango.

Utilizando este ejemplo se pueden generar las distintas clases aplicables al módulo en cuestión, luego, se procede a ingresarle al módulo un valor de cada clase.

Las pruebas de caja blanca son mucho mas 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 el, esto que

parece complicado es mas aún cuando el programa contiene código de difícil alcance, como por ejemplo manejadores de errores o "código muerto".
Entiéndase por código muerto a aquellas funciones y/o procedimientos que se han incluido por encontrarse en recopilaciones pero que estas nunca son ejecutadas por el programa, estas funciones no necesariamente deberán ser removidas pero si probadas por si algún día en revisiones futuras son incluidas.
Para los módulos que no poseen condiciones basta con ejecutar una vez el programa para asegurar una cobertura total.

Con respecto a la cobertura en bucles el tema es un poco mas delicado, a simple vista un bucle no es mas que un salto condicional que se repite hasta que se cumpla o deje de cumplirse una o mas condiciones, en teoría esto es simple, pero en la práctica son una fuente inagotable de versátiles errores, que en su gran mayoría suelen ser catastróficos.

3.3 Pruebas de caja blanca.

Para realizar las pruebas de caja blanca se siguen los pasos descritos en Capitulo 2. Epígrafe 2.4.1 Análisis de complejidad.
Porción de código para insertar una consulta oftalmológica en BD.

1. Numeración de las líneas de código.

```
public void Execute(AdoHelper helper, EPConsultaEspecializadaOftalmologica entity)
{ 1
   IDbConnection conn = helper.GetConnection(
                                        ConfigurationManager.ConnectionStrings["BQ"].ConnectionString); 2
   conn.Open(); 2
   IDbTransaction transaction = conn.BeginTransaction(); 2
   EPConsultaEspecializadaRepositorio consultaEspecializadaRepositorio =
                                        new EPConsultaEspecializadaRepositorio(); 2
```

```
EPExamenOftalmologicoEstructuraOjoRepositorio examenOftalmologicoEstructuraOjoRepositorio =
                                        new EPExamenOftalmologicoEstructuraOjoRepositorio(); 2
AntecedentePatologicoOftalmologicoRepositorio antecedentePatologicoOftalmologicoRepositorio =
                                        new AntecedentePatologicoOftalmologicoRepositorio(); 2
EPOtrosExamenesOftalmologicosRepositorio otrosExamenesOftalmologicosRepositorio =
                                        new EPOtrosExamenesOftalmologicosRepositorio(); 2
CirugiaAnteriorConsultaRepositorio cirugiasAnterioresConsultaRepositorio =
                                        new CirugiaAnteriorConsultaRepositorio(); 2
try 2
{
   EPConsultaEspecializada consulta = entity; 2
   consultaEspecializadaRepositorio.Adicionar(transaction, ref consulta); 2

   foreach (EPExamenOftalmologicoEstructuraOjo var in entity.ExamenOftalmologicoEstructuraOjo) 3
   {
      var.ExamenOftalmologicoAnatomia.IdPersona = entity.DatosConsulta.IdPersona; 4
      var.ExamenOftalmologicoAnatomia.NumeroConsulta = entity.DatosConsulta.NumeroConsulta; 4
      EPExamenOftalmologicoEstructuraOjo tmp = var; 4
      examenOftalmologicoEstructuraOjoRepositorio.Adicionar(transaction, ref tmp); 4
   }
   foreach(AntecedentePatologicoOftalmologico var in entity.AntecedentesPatologicosOftalmologicos)5
   {
      var.IdPersona = entity.DatosConsulta.IdPersona; 6
      var.NumeroConsulta = entity.DatosConsulta.NumeroConsulta; 6
      AntecedentePatologicoOftalmologico tmp = var; 6
      antecedentePatologicoOftalmologicoRepositorio.Adicionar(transaction, ref tmp); 6
   }
   foreach (EPOtrosExamenesOftalmologicos var in entity.OtrosExamenesOftalmologicos) 7
   {
      var.ExamenesOftalmologicos.IdPersona = entity.DatosConsulta.IdPersona; 8
      var.ExamenesOftalmologicos.NumeroConsulta = entity.DatosConsulta.NumeroConsulta; 8
```

```
            EPOtrosExamenesOftalmologicos tmp = var; 8
            otrosExamenesOftalmologicosRepositorio.Adicionar(transaction, ref tmp); 8
        }
        foreach (CirugiaAnteriorConsulta var in entity.CirugiasAnterioresConsulta)  9
        {
            CirugiaAnteriorConsulta cirugiasAnterioresConsulta = var; 10
            cirugiasAnterioresConsulta.IdPersona = entity.DatosConsulta.IdPersona; 10
            cirugiasAnterioresConsulta.NumeroConsulta = entity.DatosConsulta.NumeroConsulta; 10
            cirugiasAnterioresConsultaRepositorio.Adicionar(transaction, ref cirugiasAnterioresConsulta); 10
        }
        transaction.Commit(); 11
    }
    catch (Exception) 12
    {
        transaction.Rollback(); 13
        throw;
    }
    finally
    {
        conn.Close(); 14
    }
}15
```

2. Construir el grafo de flujo asociado al código representado anteriormente. **Imagen 2**

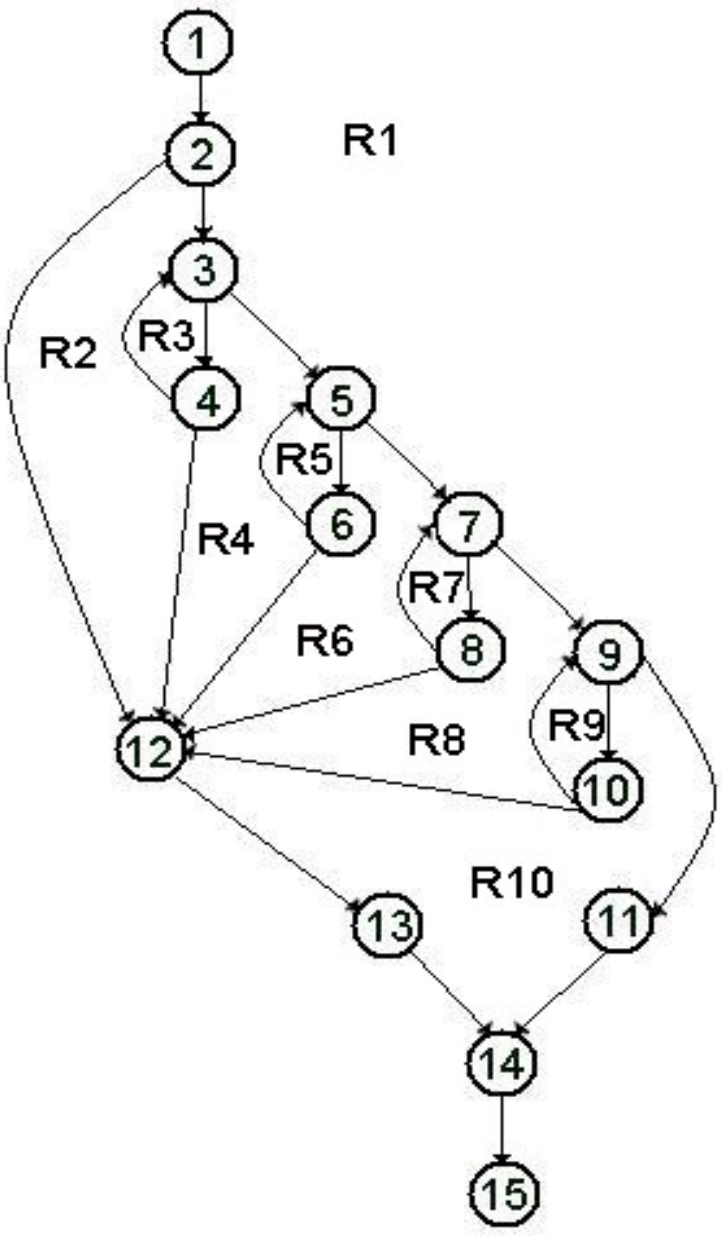


Imagen 2. Grafo de complejidad para el código anterior.

Aplicación de las fórmulas para calcular complejidad ciclomática. Para que el resultado este correcto debe ser el mismo para las tres fórmulas.

1 – V (G) = (A – N) + 2.

2 – V (G) = P + 1.

3 – V (G) = R.

Siendo “A” la cantidad total de aristas del grafo, “N” la cantidad total de nodos del grafo, “P” cantidad total de nodos predicados (son los nodos que condicionan el camino), “R” cantidad de total regiones (áreas cerradas del grafo que se encuentran enmarcadas entre un conjunto de nodos y aristas) del grafo.

Aplicando fórmula 1.

V (G) = (A – N) + 2

V (G) = (23 - 15) + 2

V (G) = 10.

Aplicando fórmula 2.

V (G) = P + 1.

V (G) = 9 + 1

V (G) = 10.

Aplicando fórmula 3.

V (G) = R

V (G) = 10.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 10, esto significa que existen diez posibles caminos por donde el flujo puede circular.

Este algoritmos es un caso particular, 5 de los caminos condicionados está dado por la presencia de una posible excepción, que solo tomaría este camino en caso de ocurrir algún error. Este algoritmo está conformado por varias sentencias repetitivas (bucles) que se ejecutan una tras la otra, de esta manera solo es necesario probar que estos bucles se ejecutan completamente, garantizando una cobertura total con una prueba, donde se introduzca la información necesaria para que todos sean ejecutados. El camino básico que será objetivo de prueba es: 1-2-3-5-7-9-11-14-15, este camino garantiza que se ejecuten todos los bucles, en caso de ocurrir un error perteneciente a las instrucciones internas, se iría por el camino del tratamiento de errores.

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 generales de la consulta (fecha hora, id de la persona, motivo de consulta), antecedentes patológicos (no son obligatorios pero necesarios para el caso de prueba), antecedentes oftalmológicos (caso anterior), exámenes por estructura del ojo (caso anterior), exámenes oftalmológicos, valoración (impresión diagnóstica, conducta a seguir).

Condición de ejecución: Todos los datos serán entrados.

Entrada:

Datos generales de la consulta.

Id persona = 15, fecha = 20/06/2007, hora= 04:59:55 PM, motivo de consulta = “Motivo de Consulta prueba”.

Antecedentes Patológicos (colección de antecedentes): “Asma”, “Cardiopatía congénita”.

Antecedentes Oftalmológicos (colección de antecedentes): “Miopía”, “Hipermetropía”.

Exámenes por estructura del ojo (colección de exámenes): ojo derecho: “Ectropión”, ojo izquierdo:” Triquiasis”

Exámenes oftalmológicos: 10.0 para todos los campos del ojo derecho y 20.0 para todos los capos del ojo izquierdo.

Valoración. Impresión Diagnóstica: “Impresión diagnóstica prueba”, conducta a seguir: “Quirúrgica”.

Resultados esperados: Se debe mostrar en el historial de consulta la nueva consulta añadida y activarse la opción de crear anuncio en caso que la conducta a seguir sea “Quirúrgica”.

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

Luego de aplicar el casos de pruebas, 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 la aplicación de este tipo de prueba se usará el caso de uso “Gestionar anuncio operatorio”. Una vez concluida la consulta el especialista determina la conducta a seguir, si esta es quirúrgica se precede a la creación del anuncio operatorio.

Objetivo del test:

Validar el caso de uso Gestionar anuncio operatorio.

Las pruebas realizadas a este caso de uso son:

- Crear anuncio operatorio oftalmológico.
- Buscar anuncio operatorio oftalmológico.
- Modificar anuncio operatorio oftalmológico.
- Posponer anuncio operatorio oftalmológico.
- Suspender anuncio operatorio oftalmológico.

Flujo central

- Una vez terminada la consulta, el especialista determina la conducta a seguir, si esta es quirúrgica, después de ser guarda, el sistema muestra la opción de crear anuncio operatorio oftalmológico.
- El especialista llena los datos del anuncio operatorio oftalmológico.
- El especialista selecciona la opción “Ver Planificación Personal” para asignar el turno quirúrgico.
- El especialista realiza la búsqueda del lente.
- El especialista completa los datos del anuncio operatorio teniendo en cuenta la planificación quirúrgica y el lente escogido.
- El especialista pulsa el botón “Aceptar”

Pre condiciones

Tienen que estar presente en la base de datos.

- Registro de personas.
- Registro de funcionarios.
- Servicios.

- Sub servicios.
- Técnicas quirúrgicas.
- Lugar de colocación del lente.
- Tipo de lente.
- Causas de suspensión.
- Fórmulas para el cálculo de refracción.

Sección: Crear Anuncio Operatorio					
Clases válidas	**Clases no válidas**	**Resultados de la prueba.**	**Resultados de la prueba.**	**Observaciones**	**Cumplimento**
El especialista selecciona la opción “Crear anuncio operatorio” del menú principal de trabajo.	No se muestra la interfaz del anuncio operatorio.	Muestra la interfaz del anuncio operatorio.	Satisfactoria	La operación se realizo correctamente	100%
El especialista selecciona la opción “Ver Planificación Personal”.	El sistema no muestra la planificación personal del especialista.	El sistema muestra la planificación personal del especialista.	Satisfactoria	La operación se realizo correctamente	100%
El especialista realiza la búsqueda del lente.	El sistema no muestra la interfaz de los lentes.	El sistema muestra la interfaz de los lentes con la	Satisfactoria	El proceso de mostrado no es de forma instantánea,	100%

		refracción esperada para ambas cámara y la disponibilidad.		debido al número de operaciones realizadas por el algoritmo para el cálculo de la refracción esperada.	
El especialista recoge los datos específicos de oftalmología y presiona el botón "Guardar".	El sistema no registra los datos.	El sistema registra los datos.	Satisfactoria	Si no se pudo registrar la información el sistema muestra el siguiente mensaje de error: "**Error: El sistema no puede guardar el registro; existen datos incorrectos"**	100%

Tabla 53 "Sección: Crear Anuncio Operatorio"

Sección: Buscar Anuncio Operatorio					
Clases válidas	**Clases no válidas**	**Resultados de la prueba.**	**Resultados de la prueba.**	**Observaciones**	**Cumplimiento**
El usuario selecciona la opción "Buscar anuncio	El sistema no muestra la interfaz correspondiente a la búsqueda de	El sistema muestra la interfaz correspondiente a la búsqueda de	Satisfactoria	El sistema muestra correctamente la interfaz correspondiente	100%

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

Tabla 54 "Sección: Buscar Anuncio Operatorio"

Sección: Modificar Anuncio Operatorio					
Clases válidas	**Clases no válidas**	**Resultados de la prueba.**	**Resultados de la prueba.**	**Observaciones**	**Cumplimiento**
El especialista selecciona la	No se muestra la interfaz del	Muestra la interfaz del	Satisfactoria	La operación se realizo	100%

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

Tabla 55 “Sección: Modificar Anuncio Operatorio”

Sección: Posponer Anuncio Operatorio					
Clases válidas	**Clases no válidas**	**Resultados de la prueba.**	**Resultados de la prueba.**	**Observaciones**	**Cumplimiento**
El especialista selecciona la opción ”Posponer	No se muestra la interfaz correspondiente a la “Búsqueda	Se muestra la interfaz correspondiente a la “Búsqueda	Satisfactoria	Se siguen los mismos pasos de la sesión Buscar Anuncio	100%

anuncio operatorio”	del Anuncio”.	del Anuncio”.		Operatorio.	
El especialista pospone el anuncio operatorio modificando la fecha y agregando la causa; pulsa el botón “Aceptar”.	El sistema no guarda los datos.	El sistema guarda los datos.	Satisfactoria	Después de salir de la sesión buscar anuncio el sistema muestra la interfaz de crear anuncio pero ahora es para modificar. Una vez pospuesto el anuncio se guardan los datos, sesión modificar anuncio operatorio.	100%

Tabla 56 “Sección: Posponer Anuncio Operatorio”

Sección: Suspender Anuncio Operatorio					
Clases válidas	**Clases no válidas**	**Resultados de la prueba.**	**Resultados de la prueba.**	**Observaciones**	**Cumplimiento**
El especialista selecciona la opción	No se muestra la interfaz correspondiente a la “Búsqueda	Se muestra la interfaz correspondiente a la “Búsqueda	Satisfactoria	Se siguen los mismos pasos de la sesión Buscar Anuncio	100%

"Suspender anuncio operatorio".	del Anuncio".	del Anuncio".		Operatorio.	
El especialista suspende el anuncio operatorio y agrega la causa; pulsa el botón "Aceptar".	El sistema no guarda los datos.	El sistema guarda los datos.	Satisfactoria	Después de salir de la sesión buscar anuncio el sistema muestra la interfaz de crear anuncio pero ahora es para modificar. Una vez suspendido el anuncio se guardan los datos, sesión modificar anuncio operatorio.	100%

Tabla 57 "Sección: Suspender Anuncio Operatorio"

En este capítulo se abordaron los métodos de pruebas usados en el desarrollo de la aplicación. Se realizaron pruebas de caja blanca y caja negra, demostrando su adecuado funcionamiento. Estas acciones realizadas durante la implementación tienen el objetivo de asegurar que el sistema implementado sea correcto y tenga calidad, para así lograr mayor satisfacción en el cliente final.

CONCLUSIONES

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

- Se implementó una aplicación para la informatización del proceso quirúrgico oftalmológico, capaz de gestionar los altos volúmenes de información generada, con una interfaz orientada al usuario. **(Ver anexo 9)**
- En la implementación de dicha aplicación se utilizaron los patrones de diseño establecidos en el análisis. Propuestos por los arquitectos del sistema.
- Se llevó a cabo una profunda investigación de las principales tecnologías utilizadas en el desarrollo de aplicaciones web.
- El estudio de los estándares de codificación fue de gran utilidad en la implementación, sirvió para aprender nuevos método y formas de tener organizado el código, lo cual ayuda en la comunicación entre desarrolladores, y permite una temprana detección de errores, facilitando las futuras actualizaciones mantenimientos o iteraciones para mejorar la aplicación.
- Se realizaron pruebas de caja blanca y caja negra, demostrando su adecuado funcionamiento. Estas acciones realizadas durante la implementación tienen el objetivo de asegurar que el sistema sea correcto y tenga calidad, para así lograr mayor satisfacción en el cliente final.

De esta forma se le da cumplimiento a los objetivos trazados para la elaboración del trabajo de diploma obteniendo la implementación de un sistema informático que permita mejorar los procesos de gestión de información en los hospitales oftalmológicos del país.

RECOMENDACIONES

- Continuar con el desarrollo del sistema, incorporarle los demás servicios oftalmológicos como:
  - Retina.
  - Neuroftalmología
  - Córnea.
  - Oculoplastia.
  - Baja Visión.
  - Glaucoma
  - Oftalmología Pediátrica.

- Someter el sistema a pruebas de calidad para la validación del mismo.
- Desplegar el sistema en hospitales oftalmológicos del país.

REFERENCIA BIBLIOGRAFÍCA

[1] MINREX, C. La informatización en Cuba, 2004. [5 de marzo 2007]. Disponible en: http://www.cubaminrex.cu/Sociedad_Informacion/Cuba_SI/Informatizacion.htm

[2] ACTUALSOFT! Software Profesional, 2005. [10 de Abril 2007]. Disponible en: http://www.actualsoft.com.ar/plus.htm

[3] VISIONDAT. Software de Oftalmología, 2006. [10 de Abril 2007]. Disponible en: http://www.visiondat.com/index.php?mod=visiondat

[4] VALLE, J. G. Definición arquitectura cliente servidor, 2005. [5 de Mayo 2007]. Disponible en: http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml

[5] VALLE, J. G. Definición arquitectura cliente servidor, 2005. [5 de Mazo 2007]. Disponible en: http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml

[6] ADR INFOR S.L. *ASP.NET,* 2007. [12 de Marzo 2007] Disponible en: http://www.adrformacion.com/cursos/aspnet2/leccion3/tutorial1.html

[7] CORPORATION, M .NET Framework, 2006. [20 de Marzo 2007]. Disponible en: http://www.microsoft.com/downloads/details.aspx?displaylang=es&FamilyID=0856eacb-4362-4b0d-8edd-aab15c5e04f5

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

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

[10] RIZZI, I. F. M. COMPLEJIDAD CICLOMÁTICA, 2006. [25 de Marzo 2007]. Disponible en: http://www.itba.edu.ar/capis/rtis/articulosdeloscuadernosetapaprevia/RIZZI-COMPLEJIDAD.pdf

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

[12] Cig_Labs. *The Home of Groundbreaking Software Quality* Management Research, 2002. [20 de Junio 2007] Disponible en: http://www.cigitallabs.com/reso/definitions/software_testing.html

BIBLIOGRAFÍA

- ANGSHUMAN CHAKRABORTI, U. K., ROOPENDRA JEET SANDHU. Microsoft .NET Framework Professional Projects. Edit. Thomson Course Technology, 2002. 943 p.

- AUTOR ROBERT L. LAIR, J. PURE ASP.NET. Edit. Sams Publishing. 2001. 613 p.

- BOOCH, G: “Object Oriented Development”, en Trans. of Soft. Eng. Vol. Se-12, Num. 2, 1986, p. 211-221.

- BHASIN, H. Asp.Net Professional Projects. Edit Thomson Course Technology. 2002. 638 p.

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

- FOGGON, D. Beginning ASP.NET 2.0 Databases: From Novice to Professional. Edit. Apress, 2006. 625 p.

- GREIFF. W. R.: “Paradigma vs Metodología”, El Caso de la POO (Parte II). Edit. Soluciones Avanzadas. 1994, p. 31-39.

- Guillermo. Tutorial para la creación de un sitio Web con autenticación mediante formulario, 2006. [5 de Junio 2007] Disponible en: http://www.elguille.info/NET/ASPNET/tutorialLogin/tutorialLogin.htm

- JEFF FERGUNSON, B. P., JASON BERES. La biblia de C#. Edit. ANAYA, 2003. 841 p.

- LANE, RECHTIN, BHANSALI, GARLAN, PERRY, CREMEN, et al. 2003. Published Software Architecture Definitions, Bibliographic Definitions [Electronic Version] extraído de http://www.sei.cmu.edu/architecture/published_definitions.html#Bibliographic El 30 de mayo del 2007.

- Microsoft Corporation. Crear aplicaciones ASP .NET seguras, 2004 [5 de Junio 2007]. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/aplic_sec.asp

- Microsoft Corporation. Tutorial de ASP.NET, 2001. [1 de Junio 2007] Disponible en: http://es.gotdotnet.com/quickstart/aspplus/doc/quickstart.aspx

- Microsoft Corporation. ASP.NET Web Matrix, 2002. [2 de Junio 2007] Disponible en: http://www.microsoft.com/argentina/msdn/capacitacion/tour/introduccion/intro.asp

- OJEDA, F. C. Visual C# .NET. ANAYA, 2002. 641 p.

- PARIHAR, M. ASP.NET. Edit. ANAYA, 1005 p.

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

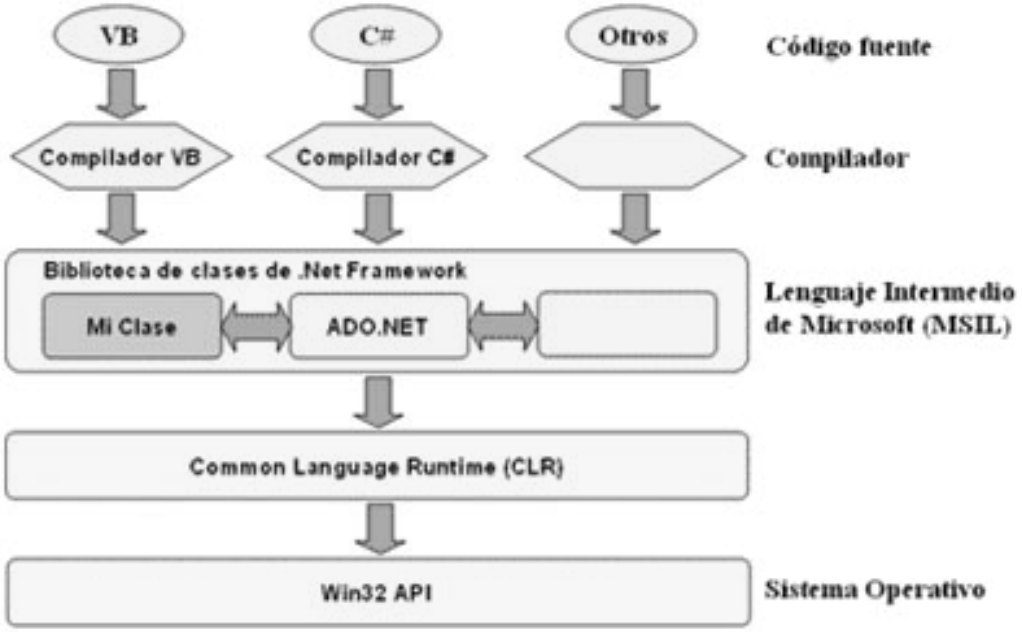
- RECIO, FRANCISCO Y PROVENCIO, DAVID: Arquitectura básica de la plataforma .Net. Descripción del Framework y sus principales componentes: Lenguajes, biblioteca de clases y CLR. En http://www.desarrolloweb.com/articulos/1328.php. Extraído el 20 de marzo del 2007.

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

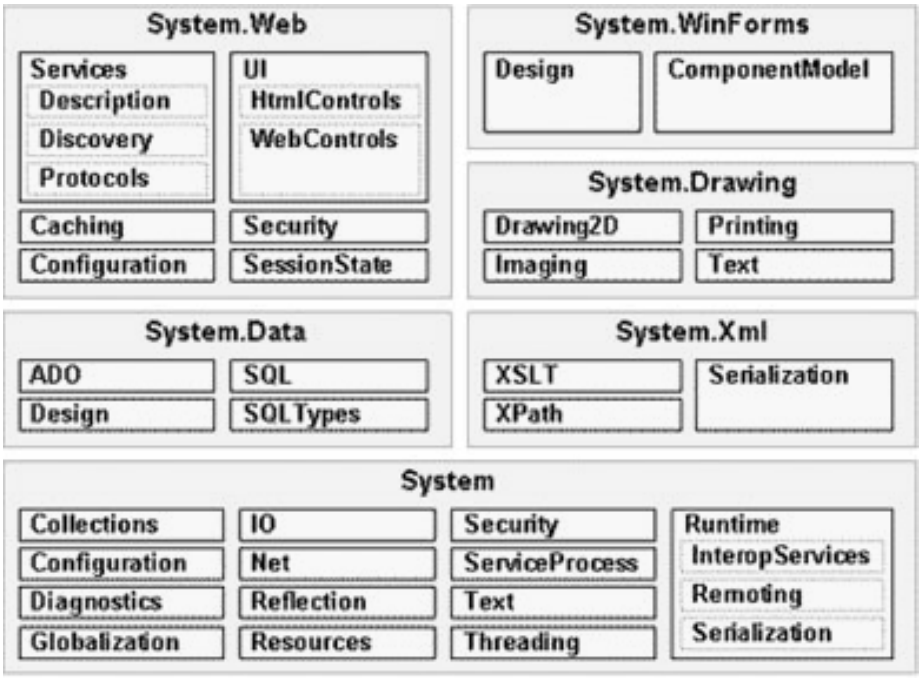
- WORLEY, S. INSIDE. Edit. Sams Publishing 2001. 736 p.

ANEXOS

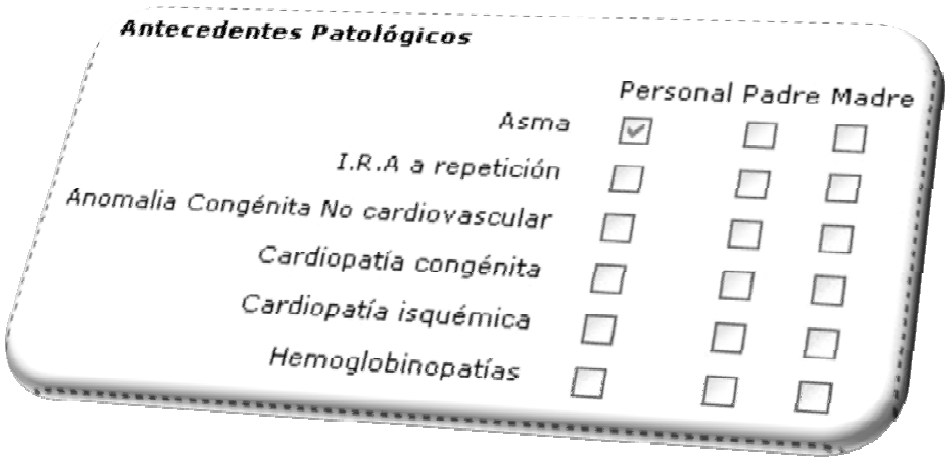
1. Arquitectura de .NET Framework.



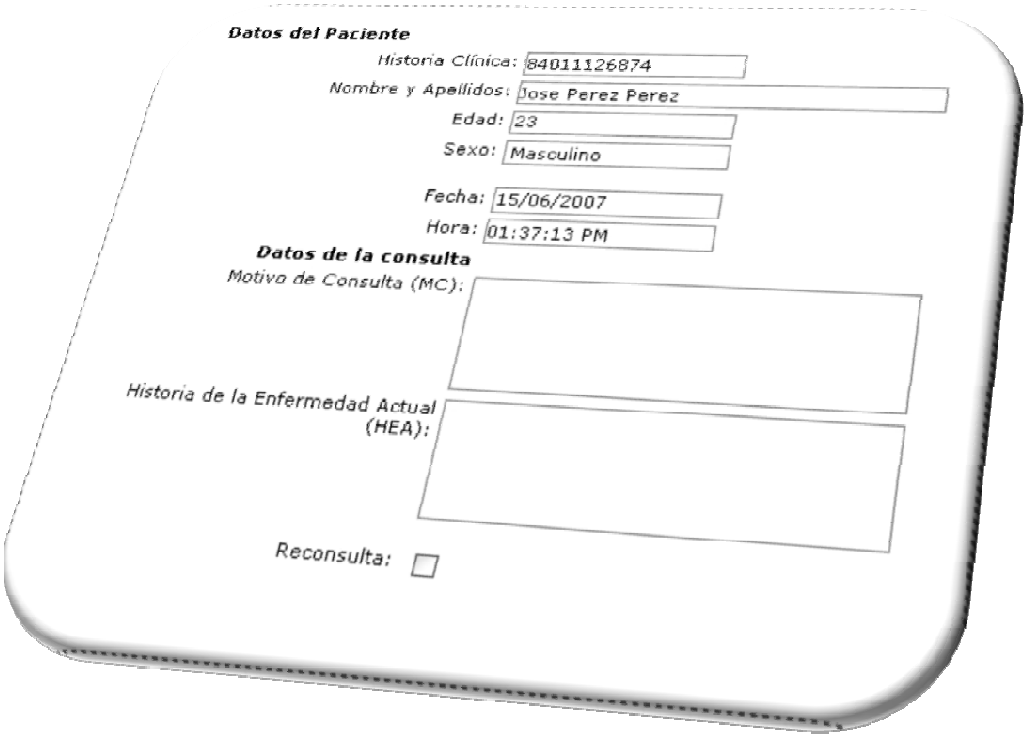
2. Biblioteca de Clases de .NET Framework.



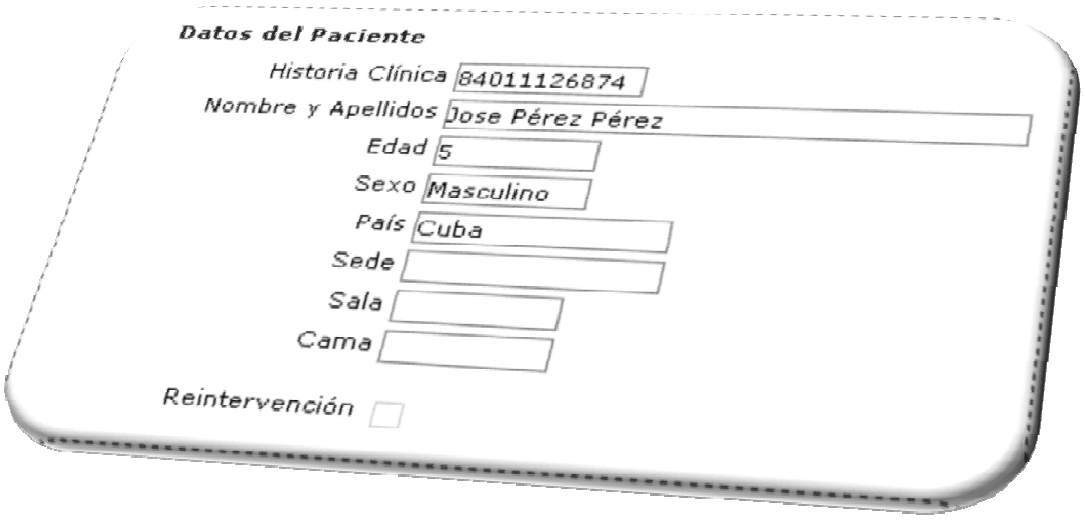
3. Antecedentes Patológicos



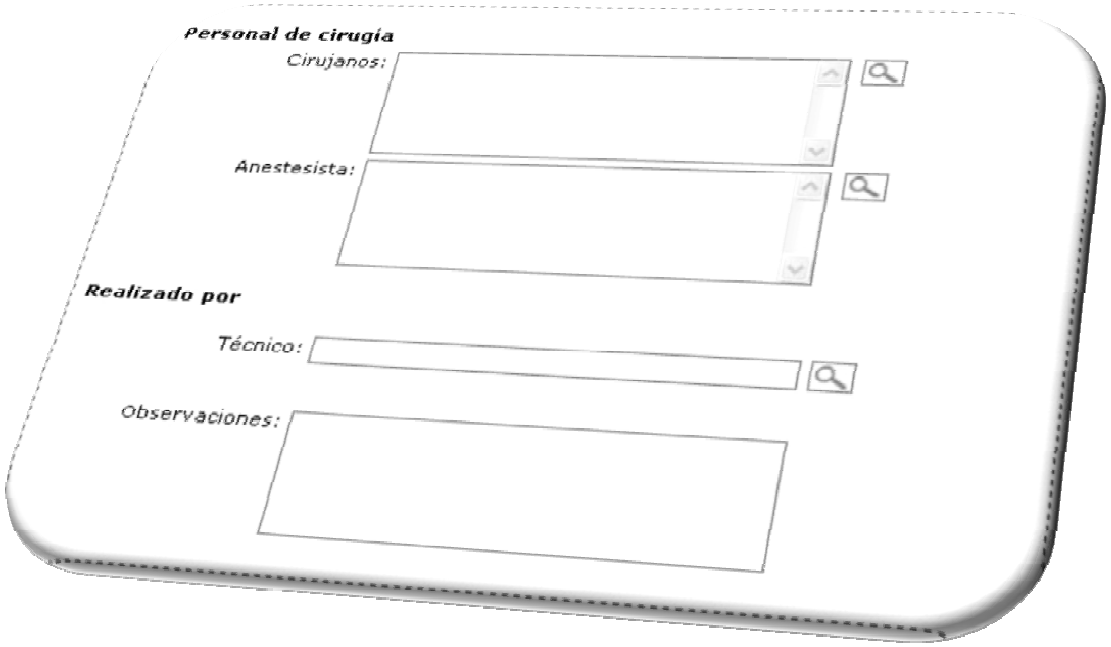
4. Datos generales de la consulta.



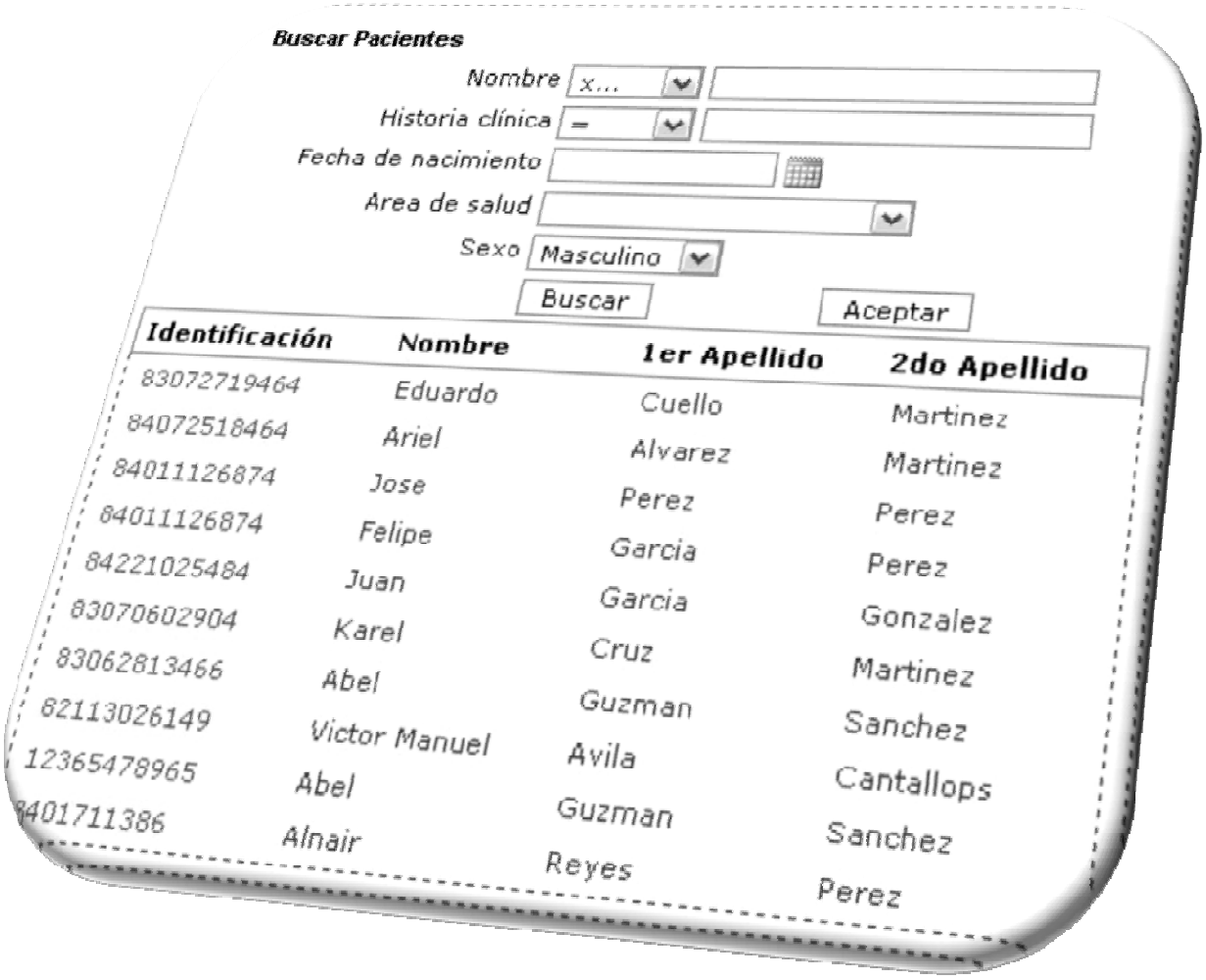
5. Datos del paciente (Anuncio operatorio).



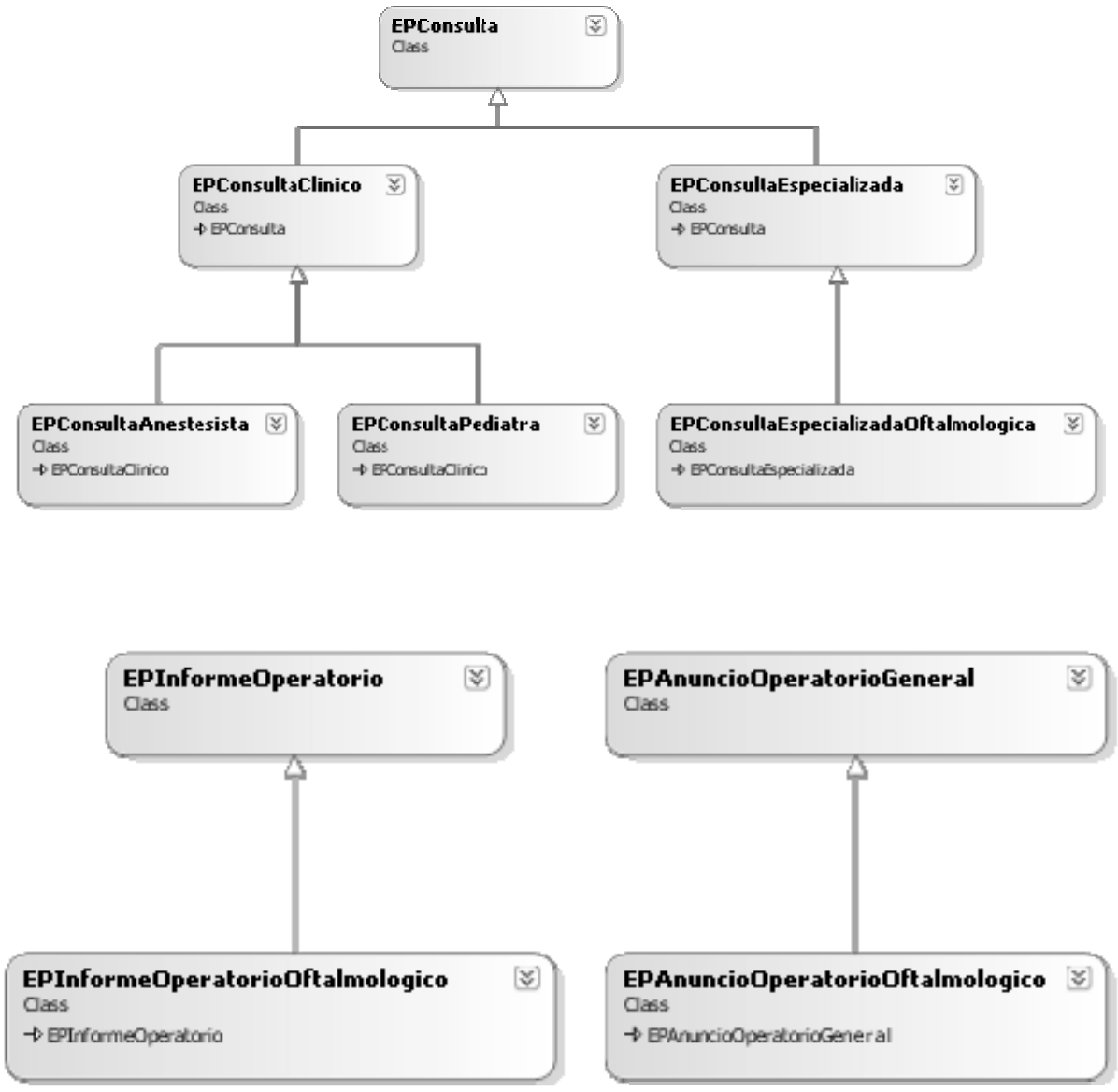
6. Personal de cirugía (Anuncio operatorio).



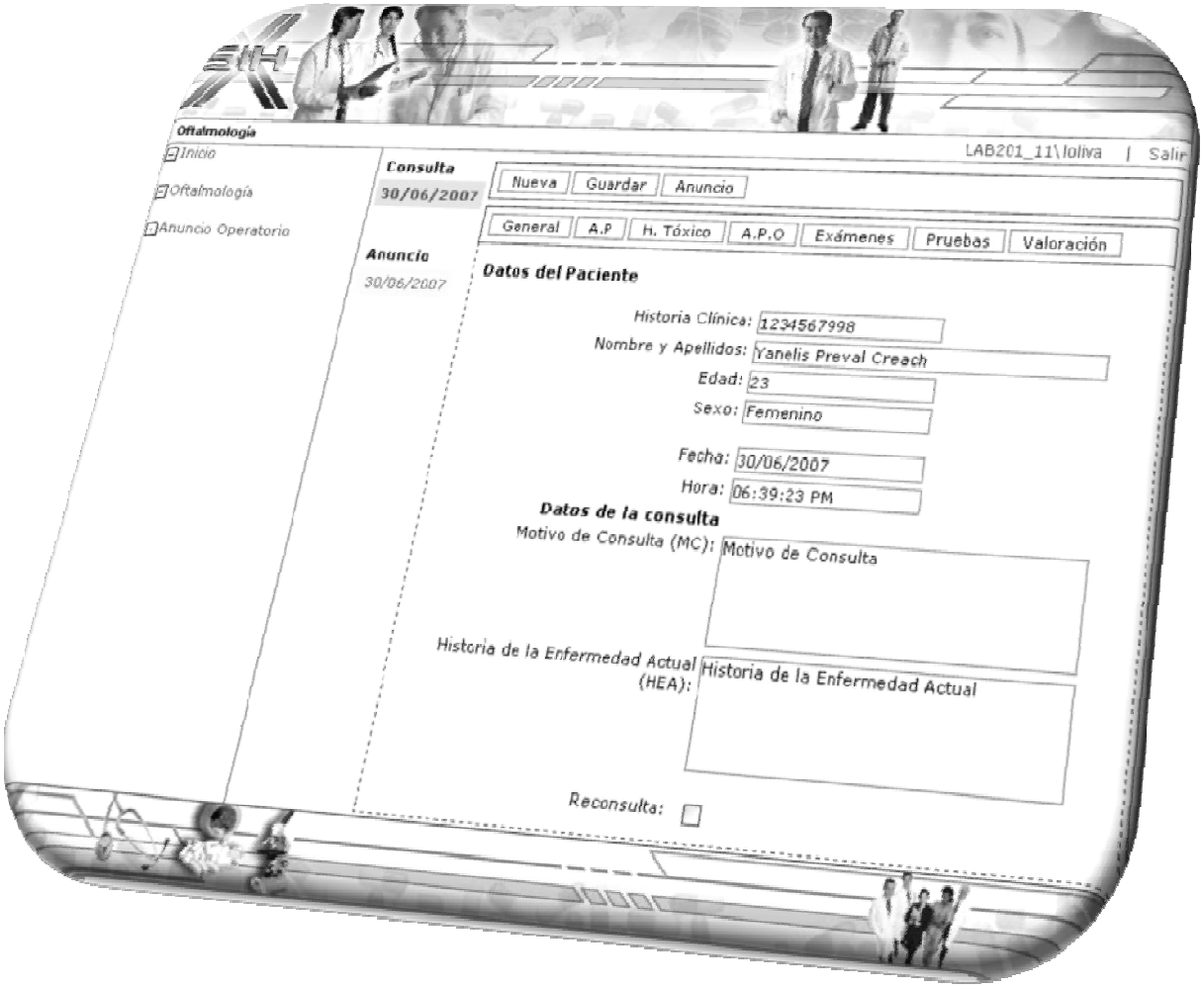
7. Buscar pacientes.



8. Diagrama de clases de negocio.



9. Interfaz gráfica (Consulta Oftalmológica).



GLOSARIO

BD: Base de Datos es un conjunto exhaustivo no redundante de datos estructurados organizados, independientemente de su utilización y su implementación en máquina accesibles en tiempo real y compatibles con usuarios concurrentes, con necesidad de información diferente y no predicable en tiempo.

CPU: Es la Unidad Central de Proceso, ó simplemente el procesador. Es el componente en una computadora digital que interpreta las instrucciones y procesa los datos contenidos en los programas de computadora.

ECMA: Organización internacional encargada de establecer estándares para la comunicación y la información, debe sus siglas a (*European Computer Manufacturers Association)*

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

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

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

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

HL7: (Health Level Seven) es una organización sin fines de lucro que desarrolla estándares para minimizar las incompatibilidades entre sistemas de información en salud, permitiendo la interacción y el intercambio productivo de datos entre aplicaciones heterogéneas, independientemente de su plataforma tecnológica o de su lenguaje de desarrollo.

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

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

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

SGBD: Sistema Gestor de Base de Datos es un conjunto de programas que permiten crear y mantener una Base de Datos, asegurando su integridad, confidencialidad y seguridad.

SGML: "Standard Generalized Markup Language" o "Lenguaje de Marcación Generalizado". Consiste en un sistema para la organización y etiquetado de documentos. La Organización Internacional de Estándares (ISO) ha normalizado este lenguaje en 1986.

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

NpgSQL: Interfaz de Programación de Aplicación (API).Proveedor de datos para el servidor de bases de datos PostgreSQL