Universidad de las Ciencias Informáticas

Facultad # 7

Título: Implementación del módulo Archivo del Sistema de Gestión Hospitalaria.

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

Autores: Dayana Pupo Palacio.

Anier Suárez Pérez.

Tutor: Ing. Paúl Pérez Zurita.

Asesor: Ing. Luis Mariano Hernández.

Ciudad de La Habana, Julio de 2007.

"Año 49 de La Revolución"

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los 5 días del mes de julio del año 2007.

Dayana Pupo Palacio. Anier Suárez Pérez. Ing. Paúl Pérez Zurita.

_____ _____ _____

"Las cadenas de la esclavitud solamente atan las manos: es la mente lo que hace al hombre libre o esclavo."

Franz Grillparze.

Agradecimientos

De Dayana:

A mi hermano, que sin su apoyo y aprobación no me hubiese lanzado a los brazos de un proyecto desconocido, que hoy es toda una universidad.

A mis padres queridos, por su dedicación y preocupación, por prepararme para la vida. Los quiero mucho.

A mis tías Yamila, Nerys, Maribel y Soledad; por ser mi apoyo, en los mejores y peores momentos de mi carrera y vida universitaria.

A mi tío Rauli, que ha sido como un segundo padre para mí.

A mi queridísimo Humberto, que junto a su familia siempre ha estado atento y dispuesto a ayudarme en lo que necesitase, dándome siempre todo su amor.

A mis profesores, que a lo largo de estos 5 años de vida universitaria han contribuido con mi formación profesional.

A Pedrito, que estuvo a mi lado en los últimos días incondicionalmente.

A la Lic. Margarita Mulet Cartaya, por su apoyo e interés hacia mi tesis y su cooperación en los últimos momentos, que fueron los más difíciles.

A todos mis compañeros de proyecto, que tanto me ayudaron y apoyaron durante la realización de este trabajo, a Alain, a Mainoldis, a Ariesky, a Leonexy y a muchos otros más.

A todos, muchas gracias.

AGRADECIMIENTOS

De Anier:

A mis padres, por su apoyo incondicional.

A Leonexy, a Ariesky, a Alain y a todos los demás amigos que colaboraron con la realización de este trabajo de diploma.

A todos los profesores que contribuyeron con mi formación profesional.

A nuestro tutor, que tanta disposición e interés mostró siempre por el trabajo.

Muchas gracias.

Dedicatoria

De Dayana:

A mis padres, que se lo merecen y tanto han esperado este momento.

A mi hermano Jorge Luis, que ya no me da envidia porque soy casi independiente y puedo hacer lo que desee.

A mi Humberto, que le deseo mucha suerte porque en menos de un año estará en mi lugar y yo en su dedicatoria.

A todos mis tíos, que los quiero tanto y ocupan en mi vida un lugar muy especial.

A mi pequeña Massielita, que es mi vida y la disfruto tanto.

A mis primos, que pronto estaremos reunidos todos juntos.

A mis abuelos, que tanto quiero y recuerdo siempre con mucho cariño.

A “Proscopito”, que acaba de nacer y tiene todo un futuro por delante.

A todos mis amigos y amigas, que cada cual sabe el pedacito que le toca en mi corazón.

De Anier:

A mis padres, que siempre han estado muy orgullosos de que esté estudiando en esta universidad de la Revolución.

A mis amigos, que tanto me han ayudado, en especial al genio de Leonexy que tanto me ha inspirado.

Resumen

La gestión de la información en los hospitales y centros médicos del país se realiza de forma limitada, debido a que debe hacerse manualmente. Esto provoca que exista un escaso control del movimiento de historias clínicas, así como de su creación y eliminación en casos determinados. Lo que conlleva a que en ocasiones existan pérdidas de información, introducción de errores y que se alargue el período de actualización de la información. Es por esto que se hace necesaria la implementación de un sistema que gestione y controle toda esta información.

El trabajo de diploma que se presenta a continuación, propone como objetivo fundamental, implementar un sistema informático para la gestión de la información de los pacientes que son atendidos en los diversos centros de atención médica del país.

Para la implementación del sistema, se utilizan como tecnologías las plataformas de desarrollo .NET para la fase de implementación y MONO para el despliegue de la aplicación; como entornos de desarrollo integrados el Visual Studio 2005 y el SharpDevelop respectivamente. Para obtener la aplicación de escritorio implementada, se usó el lenguaje de programación C# y la técnica de Programación Orientada a Objetos.

Con este proyecto se espera lograr un software multiplataforma, que posibilitará el incremento de la capacidad organizativa de los departamentos de Archivo/HC de los hospitales del país así como el aumento de la calidad de la asistencia médica a pacientes que necesitan ser atendidos médicamente.

TABLA DE CONTENIDOS

Introducción

En la actualidad, el desarrollo tecnológico hace posible que la mayoría de los países cuenten con potentes sistemas informáticos que son usados para realizar las actividades cotidianas, contribuyendo así con el desarrollo de sus sociedades.

Cuando se habla de desarrollo informático, es necesario destacar el uso de las tecnologías como una de las alternativas que ha permitido generar grandes avances en la ciencia en la mayoría de los países, lográndose con su globalización, introducir las tecnologías informáticas con nuevos paradigmas como la multimedia, la hipermedia, el Internet entre otras ventajas y desventajas que en menos de una década han renovado las viejas costumbres del papel y el lápiz.

El futuro de la humanidad dependerá en gran medida del potencial humano, de la gestión de la producción y de los conocimientos que se alcancen. La informática en sus diferentes manifestaciones, tiene asegurado un papel protagónico en este futuro. Cuba se plantea su utilización sobre principios éticos sostenibles.

La informatización del Sistema Nacional de Salud Pública está dada por el conjunto de métodos, técnicas y actividades dirigidas al manejo de la información en la salud, la cual comprende la información sobre el estado de salud de la población y la información en general para la toma de decisiones, operativas y estratégicas.

Muchas instituciones cubanas, en los últimos 20 años, han desarrollado sistemas dirigidos a lograr un determinado nivel de informatización en la salud. Estas soluciones han carecido de integración y de una definición generalizable, aparte de que no existían los recursos tecnológicos necesarios para su ejecución en el Sistema Nacional de Salud Pública.

Ya en 1997, se logra una primera estrategia de informatización, como respuesta del sector de la salud a los lineamientos estratégicos para la informatización de la sociedad cubana, con la finalidad de coordinar esfuerzos para el desarrollo de este proceso en el Sistema Nacional de Salud Pública.

En estos momentos, se trabaja íntegramente en el desarrollo de un grupo de aplicaciones básicas para la informatización del sector de la salud. En su desarrollo e implementación participan diferentes empresas del Ministerio de la Informática y Comunicaciones como Desoft, Softel, PcMax, Sys, INFOMED, CEDISAP, la Universidad de las Ciencias Informáticas (UCI) y las Direcciones Nacionales del Ministerio de Salud Pública implicadas directamente en los primeros productos.

En virtud de los alcances y desarrollos actuales de los Sistemas Informáticos Hospitalarios, no es aceptable la improvisación en el desarrollo de los mismos, en las organizaciones de la Salud.

Los sistemas de uso habitual que permiten la comunicación, intercambio y procesamiento de datos, información y conocimientos, en una Institución de Salud; tales como: los distintos registros en papel (Historias Clínicas, Certificados, Informes, Notas, etc.), el contacto persona a persona, etc. Todo esto está comprendido en las normas, reglas, actitudes y condiciones que permiten un funcionamiento adecuado, en tiempo y forma, del equipo de salud para prestar atención y cuidado a la salud del paciente.

Desde los inicios de la medicina, se tuvo la necesidad de llevar un registro escrito, de la HC del paciente, si bien, este siempre se realizó en papel, hoy en día por varios motivos, surge la necesidad de cambiar de medio de almacenamiento de la información y cada día que pase, será mas inminente el abandono del papel y el paso al registro médico electrónico.

Actualmente se acude a la transición de la HC tradicional en papel, hacia su sustitución o coexistencia con otros soportes magnéticos o digitales, que permiten almacenar y procesar gran cantidad de datos.

Es la HC, el elemento esencial de acreditación por parte del médico, de su conducta con el paciente en todo momento, al reflejar toda la información relacionada con la asistencia dispensada al propio paciente.

La HC, es el testimonio más objetivo de la calidad o de la falta de calidad del trabajo médico. Este registro médico tiene como finalidad, recoger datos del estado de salud del paciente, con el objetivo de facilitar la asistencia sanitaria. El motivo que conduce al médico a iniciar la elaboración de la HC y a continuarla a lo largo del tiempo, es el requerimiento de una prestación de servicios sanitarios por parte del paciente.

El fin principal de la HC es facilitar la asistencia sanitaria del ciudadano, recogiendo toda la información clínica necesaria para asegurar, bajo un criterio médico, el conocimiento veraz, exacto y actualizado de su estado de salud por los sanitarios que le atienden.

En Cuba, el sistema hospitalario no se encuentra automatizado en su totalidad, en una de sus particularidades el departamento de Archivo/HC (Historias Clínicas).

La mayoría de las acciones que se llevan a cabo dentro de este departamento; como son: saber la cantidad de pacientes existente, tener el control de los pacientes que han sido dados de baja, entre otras; no se realizan lo más eficientemente posible.

Esto se debe a que es muy difícil obtener toda la información de un paciente a través de informes archivados en papel; como son: sus diagnósticos, observaciones, conclusiones terapéuticas y una gran variedad de documentos e información no estructurados; los cuales no cuentan con facilidades de búsqueda y de estar digitalizados no están previstos de tecnologías avanzadas para desarrollar estas acciones.

Actualmente, en los hospitales nacionales el personal tiene poca documentación acerca de la informática y esto, trae muchos problemas a la hora de trabajar con las aplicaciones, a pesar de que las mismas sean lo suficientemente factibles para la gestión de las HC, las cuales pueden ser controladas a su vez en dependencia de diversos parámetros como: nombre del paciente, número de HC, médico por el que fue atendido y fecha de realización de la misma.

A la hora de eliminar una HC, se hace muy difícil determinar si ha sido dada de baja por su condición de pasiva, pues de estas solo se eliminan las que llevan 8 años o más en el archivo, sin haber pasado antes a estado activo.

Este limitado manejo de la información conlleva a que, en ocasiones, se demore su obtención, así como su poco control; trayendo como resultado problemas en el desarrollo del proceso médico del paciente, pues de esta información depende necesariamente el médico para tratar al mismo.

En los hospitales cubanos, los trabajadores del departamento de Archivo, desempeñan un trabajo difícil con el manejo de la HC, tratando de disminuir los posibles errores que pudieran ser cometidos.

Se emplea además, mucho tiempo en las búsquedas de las HC, así como en las entradas/salidas de las mismas; por lo que automatizando los registros médicos del paciente, se facilitaría el trabajo de muchas personas, posibilitando así la calidad, eficiencia en los servicios y factibilidad en los procesos que son vitales, para que el departamento funcione de una forma rápida y correcta.

De lo anteriormente expuesto, se deriva la necesidad de crear una aplicación informática que permita automatizar el sistema hospitalario del país, en virtud de lo cual, el problema científico de esta investigación es: ¿Cómo facilitar el acceso a la información de los pacientes en los centros hospitalarios del país, mediante la automatización del departamento de Archivo/ HC?

Correspondientemente con el problema, el **objeto de estudio** está constituido por: los procesos de gestión de la información en los centros hospitalarios del país, definiéndose como **campo de acción**: los procesos de gestión de la información en el departamento de Archivo/ HC de los centros antes mencionados.

Además, se definió como **objetivo estratégico general**, implementar un sistema informático que permita la automatización de los departamentos de Archivo/ HC de todos los centros hospitalarios del país.

Las **tareas** a desarrollar para lograr un adecuado desarrollo de este trabajo y un mejor enfoque del mismo son las siguientes:

- Analizar aspectos teóricos conceptuales de los Sistemas de Informatización de Hospitales, en especial del archivo de Historias Clínicas de Pacientes.

- Estudiar acerca de los Sistemas de Informatización de Hospitales, que se utilizan en otros países y que conllevan al mejor funcionamiento del módulo de Registros Médicos del Paciente (Archivo/ HC) en los centros hospitalarios en los que se encuentran implantados los mismos.

- Investigar acerca del lenguaje de programación escogido para la implementación del proyecto.

- Implementar las funcionalidades, que permitan controlar las HC, a través de parámetros como: número de HC, fecha de realización de la misma, médico por el que fue atendido el paciente, etc.
- Montar el sistema implementado sobre una aplicación de escritorio orientado a servicios específicos que permita la actualización automática del mismo.
- Brindar máxima seguridad para el acceso a los datos en el Sistema de Informatización de Hospitales; desde los puntos de vista de ataque de intrusos al sistema y de pérdida de datos a la hora de operar con el sistema por parte del personal autorizado.

Como estructura para el trabajo se cuenta con tres capítulos, los cuales se mencionan a continuación:

El Capítulo I. Fundamentación teórica.

Este capítulo en su contenido ofrece una panorámica de los diferentes sistemas informáticos existentes en el mundo aplicados al área de la salud, así como de las tecnologías, técnicas y demás aspectos a tener en cuenta para la realización de la aplicación.

El Capítulo II. Descripción y análisis de la solución propuesta.

En el mismo se realiza una valoración crítica de la solución propuesta por los analistas y diseñadores del sistema y se describen las clases a utilizar para el desarrollo del software.

El Capítulo III. Validación de la solución propuesta.

En este capítulo se plasma la realización de las pruebas pertinentes al software.

Cada capítulo se introduce brevemente dándose a conocer los temas que se desarrollan en el mismo y finaliza con unas conclusiones, en las cuales se reflejan los resultados alcanzados.

Con este trabajo de tesis se plantea llegar a un fructífero resultado, que permita satisfacer en su mayoría, las necesidades existentes en los centros hospitalarios cubanos, específicamente desarrollando una aplicación que permita automatizar los departamentos de archivos, que contengan los registros médicos del paciente en los diferentes hospitales del país.

Capítulo I. Fundamentación teórica.

En la actualidad se encuentran desarrollados muchos de estos sistemas para lograr un mejor funcionamiento en cada una de las esferas de la sociedad, siendo el sector de la salud privilegiado en este aspecto.

Internacionalmente se han reconocido diversos sistemas de informatización de hospitales que cuentan con características muy específicas acorde a sus funcionalidades.

El HIS (Sistema de Información Integral Hospitalario) que propone la Coordinación de Informática Médica de la Universidad Autónoma de Guadalajara (CIM UAG), se robustece en su nueva versión 4.0, la cual provee los avances en los procesos de información hospitalaria.

El HIS se divide en los siguientes sistemas:

- MIS (Sistema de Información Administrativo)
- NIS (Sistema de Información de Enfermería)
- LIS (Sistema de Información de Laboratorios)
- CIS (Sistema de Información Clínica).

El Sistema de Información Clínica se adapta a las necesidades de la sección de archivo/HC y apoya fundamentalmente las funciones del médico y su entorno. Es una herramienta creada para reducir el tiempo de trabajo e incrementar la eficacia y seguridad de la recopilación, almacenamiento y consulta de la información contenida en el Expediente Clínico y las actividades relacionadas con el cuidado de la salud del paciente, así como el procesamiento de esta información para la obtención de reportes y estadísticas que apoyen a la administración del hospital.

Las características de desarrollo con que cuenta el sistema son: IDE: Delphi 7, lenguaje de programación: object pascal, componentes first class (infopower 2000). [1]

Otro sistema identificado es el MedFile 5.x que es un software de fácil uso, efectivo y con un precio accesible, diseñado para satisfacer las necesidades de archivo de HC y manejo de turnos de un consultorio o institución médica en el que se desempeñen uno o varios profesionales (hasta 200 en la versión multiusuario).

MedFile 5.x es una aplicación de escritorio, que permite crear y mantener HC electrónicas de sus pacientes en un formato especial de *base de datos*, asignar turnos para la consulta con agenda personalizada para cada médico y emitir prescripciones y órdenes médicas en forma altamente personalizadas y configurables. El sistema está implementado en el lenguaje de programación C#, sobre la plataforma MONO, obteniéndose un producto desarrollado con software libre. [2]

El SIVSA es una empresa especializada en el desarrollo e implantación de software para el sector de la salud que cuenta con más de 15 años de experiencia. Uno de los productos desarrollados por este consorcio es el Hosix-V que está diseñado utilizando una arquitectura Cliente – Servidor accesible desde clientes Web, lo que permite ser ejecutado desde cualquier navegador ya sea vía Intranet o Internet.

Abarca todas las áreas de actividad de un hospital y pretende, a través de una utilización fácil, rentabilizar los recursos existentes para organizar el trabajo desarrollado diariamente. Está compuesto por un grupo de módulos orientados a la gestión específica de cada servicio o departamento, cubriendo los aspectos más importantes de una unidad de salud.

Este sistema cuenta con una base de datos de usuarios centralizada en la cual se encuentran todos los ciudadanos y permite una gestión integral de la información y su permanente actualización. Ha sido desarrollado utilizando como herramientas y lenguaje de programación Microsoft Visual Basic para aplicaciones, además requiere una aplicación Microsoft para su uso, su precio oscila entre los $500,001 a 2, 000,000 de dólares.

Cuenta con módulos como el de Gestión de citas, Telemedicina, Seguridad, Urgencias, Hospitalización e Historia clínica, entre otros. Este último es equivalente a lo que sería en el sistema de salud pública cubano el departamento de archivo/HC. [3]

Por otra parte, el Care2x soluciona el problema de la dependencia de plataforma, acelera las mejoras y la adaptación de módulos de programas existentes. Los costos de desarrollo y el tiempo requeridos se reducen sustancialmente. No es necesario contratar a varias compañías de software diferentes. No existe desconexión del sistema durante las actualizaciones de los módulos o durante la instalación de extensiones de los módulos. Los usuarios probablemente no se darán cuenta de que se han actualizado sus módulos de trabajo en caliente.

Uno de sus módulos es el MEDOCS, que es un sistema de registro de HC tanto para los departamentos de hospitalización como de consulta externa.

Entre sus principales funcionalidades están:

- Crear y buscar archivo.
- Listar médicos clínicos y cirujanos.
- Solicitar contraseña de ingreso a la aplicación para los médicos.
- Planificar las actividades para médicos clínicos y cirujanos, entre otros.
- Listar de forma rápida el plan de actividades de cada médico.
- Interactuar a través de foros de discusión de casos, noticias y libreta de apuntes. [4].

C-DAC es una corporación productora de software que radica en la India y cuenta con una gama de productos encaminados a la automatización de procesos no sólo del área de salud, uno de los software desarrollados por esta empresa es el C-DAC's Sushrut, que es un sistema de información hospitalario desarrollado con el objetivo de aerodinamizar el flujo del tratamiento de un paciente en el hospital, mientras que permite que los doctores y el otro personal se realicen a su capacidad máxima, en una manera optimizada y eficiente. Este SIH implementado por C-DAC contiene varios módulos los cuales aseguran los beneficios obtenidos a través de la tecnología y para la salud. El sistema utiliza una red de computadoras para reunir procesos, y manejar toda la información administrativa del hospital para satisfacer los requisitos funcionales de los usuarios. Este sistema ha sido desarrollado utilizando una arquitectura cliente servidor, además puede ser utilizado en múltiples plataformas.

Incorpora un sistema de información clínica computarizada e integrada para mejorar la administración del hospital y el cuidado de la salud de los pacientes. Además provee un archivo médico electrónico muy

preciso de los pacientes, incluyendo un almacén de datos (data warehouse) del cual los archivos pueden ser utilizados para requerimientos estadísticos y para la investigación. [5]

Nacionalmente, especialistas de la industria cubana del software han desarrollado varios productos informáticos de gran beneficio para la salud pública que en estos momentos se encuentran aplicados en varios centros asistenciales del país.

El Proyecto PHRplus (Partners for Health Reform plus), con la asistencia del USAID (Agencia de Estados Unidos para el Desarrollo Internacional, por sus siglas en inglés) son los desarrolladores del Sistema Integrado de Gestión hospitalaria Galenhos (R), que ha sido diseñado con el propósito de apoyar a los establecimientos de salud en el correcto registro de información, clínica o administrativa, y la generación de información gerencial. Como sistema hospitalario para la gestión de su información, el Galenhos cuenta con ventajas apreciables, como:

- Información estandarizada
- Bases de datos consolidadas y exportables
- Generación de reportes clínicos
- Altos estándares de seguridad informática
- Diseño modular.

El desarrollo inicial de Galenhos se enfoca en el propósito de hacer más eficiente la gestión de cinco procesos operativos críticos de un hospital: consulta externa, facturación, hospitalización, emergencia y archivo clínico. Este último es el encargado de recibir solicitudes de historias clínicas, para su búsqueda y monitorear las historias clínicas que se encuentran fuera del archivo.

Este sistema informático médico, dentro de sus características principales, contiene que no presenta una arquitectura orientada a servicios, que está desarrollado en Delphi y Visual Basic, sobre una plataforma propietaria, que la interfaz del sistema no es amigable, que los sistemas de búsqueda son estáticos y que además, no es un sistema multiplataforma.[6]

Cuando un programador se da a la tarea de desarrollar un proyecto determinado, luego de surgírsele la necesidad del mismo, lo primero que analiza es cómo lo va a realizar, qué tecnologías va a tener en cuenta y qué técnica y lenguaje de programación va a necesitar.

I.I. ¿Qué tecnologías se pueden utilizar?

Cuando se habla de tecnologías, enseguida viene a la mente una imagen de todas las nuevas tecnologías de la información y las comunicaciones existentes, pero es a las tecnologías de la programación a las que se refiere este epígrafe.

En el mundo informático, más específicamente en el de la programación, el desarrollador de software se basa en las tecnologías de la programación, para desarrollar variadas aplicaciones según las necesidades del cliente. Cada una de ellas cuenta con características propias, facilitadoras de funciones específicas a la hora de crear las aplicaciones.

Actualmente existe una gran cantidad de tecnologías usadas mundialmente en la industria del software, de las cuales se deben escoger las necesarias y más factibles para lograr el objetivo trazado.

I.I.I. ¿CON QUE PLATAFORMA SE VA A TRABAJAR?

Antes de realizar cualquier análisis, es preciso recordar que, una plataforma de desarrollo es el entorno común en el cual se desenvuelve la programación de un grupo de aplicaciones, que normalmente se relaciona directamente a un sistema operativo, pero también es posible encontrarlas ligadas a una familia de lenguajes de programación o a una Interfaz de Programación de Aplicaciones (API).

Existen diversas plataformas, cada una con un nombre que la identifica, sin dejar de mencionar las características propias de cada una de ellas con las que cuenta el desarrollador a la hora de utilizarlas.

Para la realización de la presente aplicación se tuvo en cuenta:

- .NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de

aplicaciones. Basado en esta plataforma, Microsoft intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el Sistema Operativo hasta las herramientas de mercado. La Plataforma. NET, que es una capa de software que se coloca entre el Sistema Operativo (SO) y el programador y que abstrae los detalles internos del SO.

Soporta cualquier lenguaje de programación y permite la operabilidad entre diferentes porciones de código escritos en diversos lenguajes de programación. Permite la ejecución de las aplicaciones .NET en cualquier SO de cualquier máquina que disponga de una versión de esta plataforma. En estos momentos la plataforma .NET tan solo está disponible para la familia Windows, aunque se está desarrollando una versión para Linux llamada Plataforma Mono. [7]

- MONO, que es una plataforma de desarrollo de código abierto, basada en el framework .NET. Es una plataforma para ejecutar y desarrollar aplicaciones basadas en los estándares ECMA/ISO. Mono puede ejecutar aplicaciones hechas para los framework .NET y Java.

Permite además a los desarrolladores, construir aplicaciones Linux y multiplataforma de alta productividad. La implementación de Mono cumple los estándares ECMA para C# y la infraestructura de lenguaje común (Common Language Infraestructure, CLI).

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).

Posee librerías adicionales y otras de terceras partes. Para el desarrollo de aplicaciones gráficas se incluye la librería GTK# que es un enlace 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 el Glade. Además está el Monodevelop que es un IDE (entorno de desarrollo integrado) que integra la gestión de proyectos, la construcción de aplicaciones, el depurado y toda la documentación. [8]

Teniéndose en cuenta las plataformas antes analizadas y las ventajas que muestra cada cual, se decidió trabajar con la Plataforma .NET para la implementación, debido a que cuenta con óptimas condiciones para el trabajo y el desarrollo de la aplicación, además, posee un amplio historial en el desenvolvimiento de este tipo de tareas, demostrándose así en un gran número de programadores que la han tenido en cuenta a la hora de programar y han obtenido los mejores resultados.

No obstante, se pretende lograr un producto libre (open source), teniéndose en cuenta para esto la Plataforma MONO, que contando con características homólogas a las de la Plataforma .NET, permite migrar a esta primera teniendo en cuenta el lenguaje de programación usado para desarrollar la aplicación, entre otras características que sean necesarias tener en cuenta para esto.

I.I.II ENTORNO DE DESARROLLO INTEGRADO (IDE). [1]

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de Interfaz Gráfica de Usuario (GUI). Los IDE pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. Estos entornos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, etc.

En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación de forma interactiva. Particularmente, para este proyecto se decidió utilizar el Visual Studio .NET que es la herramienta que Microsoft distribuye junto a la plataforma que permite construir y desarrollar aplicaciones .NET, siendo el mismo un software propietario al mismo tiempo que necesario para trabajar con la plataforma de Microsoft.

Este IDE es una mezcla de los diferentes entornos que Microsoft utilizaba hasta ahora (Visual Basic 6 IDE, Visual InterDev, etc.). La principal diferencia respecto a otros IDE es que utiliza exactamente el mismo entorno para todos los lenguajes incluidos en la plataforma.

[1] Para la realización de este epígrafe se consultó la Wikipedia, disponible en: http://es.wikipedia.org

Por otra parte se pudiera citar el SharpDevelop, que es un IDE compatible con ambas plataformas y permite comprobar y controlar que el código obtenido con el Visual Studio .Net pueda luego utilizarse en la Plataforma Mono, pues con él se pueden importar los proyectos creados con el software propietario antes analizado.

El SharpDevelop soporta las versiones del Framework de Microsoft y el desarrollo de interfaces, clases y proyectos en C#, además, para el completado automático de código, la aplicación incorpora sus propios parsers. Actualmente cuenta con su versión 2.0 que ya es capaz de editar directamente los proyectos que antes eran necesariamente realizados con el Visual Studio .NET.

I.I.III. ¿APLICACIÓN WEB O DE ESCRITORIO?

Actualmente hay una gran diferencia entre las posibilidades existentes a la hora de crear una aplicación de escritorio y una aplicación web, aunque es verdad que esta diferencia se ha reducido con la utilización de diversas técnicas y herramientas que han ido surgiendo con la necesidad de mejorar esta última.

Actualmente, con la masificación de Ajax y el desarrollo de plataformas como Flex u Open Lazlo, las RIA (”Rich Internet Aplications”, o aplicaciones web de comportamiento similar a las de escritorio) se están poniendo a la defensiva y tendrán que ser tomadas seriamente en cuenta.

El objetivo de estas tecnologías es disminuir las limitaciones de HTTP y construir interfaces “fluidas” y más usables que imiten la inmediatez de las aplicaciones de escritorio.

La diferencia entre la web y el escritorio, es que esta primera no mantiene su estado, lo cual quiere decir que, entre una página y otra, no hay “memoria” de las acciones efectuadas anteriormente por el usuario. Cada vez que se presiona un link, el navegador envía una petición (”request”) al servidor, quien a su vez procesa los datos enviados y responde adecuadamente. El navegador recibe y despliega esta respuesta y la comunicación termina hasta la siguiente interacción del usuario. Esta es la arquitectura que ha dado a entender Internet como una serie de “páginas” o documentos individuales, relacionados entre si por hipervínculos.

Las aplicaciones de escritorio, por el contrario, mantienen un contacto permanente entre los procesos internos del programa y lo que sucede en la interfaz de usuario. Es por esto que no requieren del paradigma de páginas de la web y tienden a ofrecer una experiencia de usuario más fluída entre una acción y otra.

Precisamente, esta última fue escogida para ser el cuerpo de este proyecto, teniéndose en cuenta la necesidad de crear una aplicación que se ajuste a las expectativas del cliente, facilitándole un mejor desempeño personal e institucional, de una manera confiable, segura y rápida. Es de máxima preocupación abarcar los más significativos detalles que puedan costar tiempo y esfuerzo, para lo cual se ha optado por ofrecer un servicio de calidad y estabilidad, asegurándose que el sistema realizado cuente con las más estrictas normas de calidad y así asegurar una solución inteligente y eficaz para el cliente.

I.II. Técnicas y Lenguajes de Programación.

Cuando se planea realizar un programa, es necesario saber cual técnica y lenguaje de programación se va a utilizar, teniendo en cuenta previamente la plataforma, el entorno de desarrollo y el tipo de aplicación que se escogieron para trabajar.

Según sus propias particularidades y características, las técnicas de programación están definidas como: Programación no Estructurada, Programación Procedimental, Programación Modular y Programación Orientada a Objetos.

A la hora de lograr una aplicación mas objetiva y que cumpla en su totalidad con los requerimientos y expectativas de un cliente, se hace la necesidad de trabajar con la última técnica de programación, que siendo orientada a objetos, resuelve algunas de las desventajas de sus antecesoras. En contraste con las otras técnicas, ahora se tiene una red de objetos que interactúan entre sí, los cuáles mantienen su propio estado.

Entrando en términos generales, cada objeto implementa su propio módulo, permitiendo por ejemplo que coexistan muchas listas y es además, responsable de inicializarse y destruirse en forma correcta. Es por esto no necesario llamar explícitamente al procedimiento de creación o de terminación.

Claramente, resaltan las ventajas que fueron apareciendo con la evolución de las técnica de programación, pero es importante destacar su utilización en los lenguajes de programación existentes en el amplio mundo de la informática. En la actualidad, se utilizan de forma general los lenguajes orientados a objetos y para esto es imprescindible el uso de esta última técnica, descartándose entonces el resto ellas.

Después de realizada la elección y considerando que es la técnica más adecuada para trabajar en la aplicación, debido a su superioridad con respecto a las otras, fue preciso escoger un lenguaje de programación que facilitara el trabajo y que además fuese compatible con la anterior técnica.

Son muchos los lenguajes de programación que podrían ser candidatos, entre los cuales se puede citar: el C, el C++, el J++ y el C#, etc., siendo este último considerado el más completo y viable para el trabajo a realizar.

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

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

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

Aunque C# forma parte de la plataforma .NET y esta es una interfaz de programación de aplicaciones; C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Aunque aún no existen, es posible implementar compiladores que no generen programas para dicha plataforma, sino para una plataforma diferente como Win32 o UNIX.

En la actualidad existen los siguientes compiladores para el lenguaje C#:

- Microsoft .NET framework SDK incluye un compilador de C#.
- Microsoft Visual C#, IDE por excelencia de este lenguaje, versión 2002, 2003 y 2005.
- SharpDevelop, es un IDE libre para C# bajo licencia LGPL, muy similar a Microsoft Visual C#.
- Mono, es una implementación GPL de todo el entorno .NET desarrollado por Novell. Como parte de esta implementación se incluye un compilador de C#.
- Delphi 2006, de Borland Software Corporation.
- dotGNU Portable.NET, de la Free Software Foundation.[9]

Analizándose la superioridad de este lenguaje, es importante mencionar que el mismo puede ser implementado lo mismo en un IDE propietario como el Visual Studio .NET que en uno libre (open source) como el SharpDevelop.

I.III. Qué usar para la conexión a la Base de Datos?

Es impensable trabajar como programador y no tener que trabajar con bases de datos. De hecho, la mayor parte del trabajo de un programador consiste en gestionar datos. Ello implica tratarlos en todos los niveles: definición y diseño, consulta, modificación y acceso a los mismos desde aplicaciones Web o de escritorio.

Debido a que el Gestor de Bases de Datos seleccionado para organizar y gestionar la información de la aplicación es el PostgreSQL, el cual ofrece un alto rendimiento en la conectividad a la base de datos en cuestión y un número de herramientas y de tecnologías innovadoras de desarrollo; y que además contiene componentes adicionales diseñados para simplificar algunas tareas; fue necesario incluir en este trabajo una librería específica para realizar las consultas necesarias a esta base de datos y que fuese compatible con el gestor antes mencionado.

Teniendo en cuenta esta necesidad se decidió utilizar la librería Npgsql, que es un proveedor de datos .NET para el servidor de bases de datos PostgreSQL. Este permite a una aplicación .NET usar un servidor

PostgreSQL para enviar y recibir datos. El mismo ha sido desarrollado usándose las especificaciones de la documentación de .NET.

En este capítulo, se analizan y describen algunos conceptos necesarios para la comprensión del trabajo y se fundamenta la utilización de los mismos en la elaboración del producto en cuestión. Además se hace un análisis de los recursos que serán utilizados a lo largo del desarrollo del sistema propuesto, argumentándose la elección final para desarrollar el trabajo adecuadamente.

Capitulo II. Descripción y análisis de la solución propuesta.

En este capítulo, se explica el diseño alcanzado como propuesta para desarrollar el sistema debidamente, los principales aspectos que se han tenido en cuenta para la implementación del mismo, teniendo presente la propuesta de los analistas y diseñadores del sistema para lograr así un producto con la mayor calidad y eficiencia requerida.

El objetivo principal ha sido detallar los principales procesos con los que se deben trabajar para lograr que una vez implementado el sistema, este satisfaga las necesidades del usuario final.

II.I. Valoración crítica del diseño propuesto por los analistas.

Los analistas y diseñadores del sistema han brindado una propuesta acorde al desarrollo que se quiere lograr, primero a la hora de implementar el sistema y luego a la hora de poner en práctica el mismo.

En el se definen correctamente las clases fundamentales que deben ser determinadas para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, mostrando una idea clara de lo que se debe implementar, conjuntamente a toda una serie de diagramas de interacción, que explican la colaboración existente entre las clases y cómo son llamados los métodos y sentencias dentro de cada una ellas, obteniendo así la información necesaria para conocer el orden de las acciones a implementar.

Es importante destacar la correcta determinación en cuanto a los patrones utilizados, que de manera general, constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, basados en la experiencia y que se ha demostrado que funcionan, permitiendo llevar a cabo la implementación clara y limpia del modulo bajo patrones como los GRASP y los GOF.

Los patrones GRASP son aplicados a las clases definidas en el diseño, distribuyendo responsabilidades entre las mismas de forma tal que no existan muchas relaciones, que no sea sobrecargada de métodos una clase en específico pudiendo acomodarlos en otras, entre otras mejoras que brinda el uso de este grupo de patrones.

Por otra parte, los 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 y 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 del patrón Fábrica Abstracta. Este tiene como principal función crear objetos sin importar en que momento y a que clase pertenecen y por lo general se implementa junto al patrón método de fabricación. [10]

Con el uso de estos patrones se permitirá implementar las entidades del negocio de forma independiente y fácil, debido a que las funciones básicas como son: selección, inserción, modificación, y eliminación estarían en el subsistema fábrica, comunicándose con dicho subsistema una clase Repositorio encargada de gestionar los procesos fundamentales y a esta una clase del negocio encargada de responder a la interfaz de usuario.

En los inicios resultó trabajoso el uso de estos patrones debido a que era necesario implementar las anteriormente mencionadas funciones básicas para cada entidad existente a partir del diseño, pero gracias al esfuerzo de un pequeño grupo de desarrolladores del grupo de desarrollo GeHos; que implementó una herramienta acorde a la funcionalidad requerida, para generar toda la capa middleware existente entre las capas de acceso a datos y negocio; no surgió una limitante que obstaculizara la tarea de construir este sistema.

II.II. Reutilización de componentes o módulos ya existentes.

Para lograr una solución apropiada al problema propuesto, o sea, la creación del sistema informático ya tratado en el capítulo anterior, es necesario la reutilización de la arquitectura de otros módulos que se encuentran ya implementados y que forman parte del Sistema de Información Hospitalaria.

Dentro del sistema se puede hacer referencia al módulo de Configuración, el cual se encarga de gestionar toda la información referente a los codificadores necesarios para el funcionamiento del módulo de archivo/HC; al módulo de Seguridad, a través del cual se gestiona la seguridad y autenticidad de la información del módulo de archivo/HC y por último, al módulo de Inscripción-Admisión, el cual se encargará de gestionar todo lo referente a las inscripciones realizadas y admisiones correspondientes a los pacientes.

Un componente ha reutilizar es la librería Npgsql, el cual es fundamental a la hora de establecer una comunicación con una base de datos PostgreSQL como fue explicado anteriormente en el epígrafe I.III del CAPITULO I.

Dentro de esta lista de módulos reutilizados se encuentra también el Hermes: Framework HL7. El mismo es una implementación confiable del estándar HL7 en su versión 2.3.1, desarrollada por un proyecto del área temática de la facultad 7 perteneciente a la Universidad de las Ciencias Informáticas, versión de HL7 más difundida a nivel mundial.

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

Contar con sus herramientas permite el envío y recepción de datos clínicos, información de pacientes y algunos reportes de laboratorio, entre los distintos sistemas de información de la salud que lo utilicen, empleando para ello un solo formato, el especificado por HL7. Reduce además, los recursos invertidos en

la negociación de las interfaces entre aplicaciones para la salud. Con su utilización se logra incrementar el prestigio del sistema de salud cubano, así como la confianza internacional en el mismo.

Permite además, el ahorro de una suma importante de dinero al país, al evitar la compra de implementaciones del estándar a empresas extranjeras.

II.III. Descripción de las nuevas clases u operaciones necesarias.

II.III.I. CLASES ENTIDADES (CE).

TABLA 1. Descripción de la CE <CE_Usuario>

Nombre: CE_Usuario	
Tipo de clase: entidad	
Atributo	**Tipo**
Numero_Identidad	Boolean
Nombre	String
Contraseña	String
Para cada responsabilidad:	
Nombre:	Get Numero_Identidad
Descripción:	Función que devuelve el número de identidad del usuario.
Nombre:	Get Nombre
Descripción:	Función que devuelve el nombre del usuario.
Nombre:	Get Contraseña
Descripción:	Función que devuelve la contraseña del usuario.
Nombre:	Set Numero_Identidad
Descripción:	Función que permite entrar el número de identidad del usuario.
Nombre:	Set Nombre
Descripción:	Función que permite entrar el nombre del usuario.

Nombre:	Set Contraseña
Descripción:	Función que permite entrar la contraseña del usuario.

2. Descripción de la CE <CE_Rol>

Nombre: CE_Rol	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_Rol	Boolean
Nombre_Rol	String
Para cada responsabilidad:	
Nombre:	Get Id_Rol
Descripción:	Función que devuelve el identificador del rol que pueda tener un usuario.
Nombre:	Get Nombre_Rol
Descripción:	Función que devuelve el nombre del rol que pueda tener un usuario.
Nombre:	Set Id_Rol
Descripción:	Función que permite entrar el id del rol del usuario.
Nombre:	Set Nombre_Rol
Descripción:	Función que permite entrar el nombre del rol del usuario.

3. Descripción de la CE <CE_Permisos>

Nombre: CE_Permisos	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_Permiso	Boolean

Nombre_Permiso	String
Para cada responsabilidad:	
Nombre:	Get Id_Permiso
Descripción:	Función que devuelve el identificador del permiso que pueda tener un usuario.
Nombre:	Get Nombre_Permiso
Descripción:	Función que devuelve el nombre del permiso que pueda tener un usuario.
Nombre:	Set Id_permiso
Descripción:	Función que permite entrar el id del permiso del usuario.
Nombre:	Set Nombre_Permiso
Descripción:	Función que permite entrar el nombre del permiso del usuario.

TABLA 4. Descripción de la CE <CE_Datos_Persona>

Nombre: CE_Datos_Persona	
Tipo de clase: entidad	
Atributo	**Tipo**
id_persona	Integer
nombre	String
primer_apellido	String
segundo_apellido	String
CI	Integer
fecha_nacimiento	Date
nombre_madre	String
nombre_padre	String
centro_trabajo	String
direcccion_centro_trabajo	String
fecha_registro	Date

factor_rh	String
telefono	Integer
trabajador_salud	Boolean
id_color_piel	Integer
id_sexo	Integer
id_municipio	Integer
Id_provincia	Integer
id_nacionalidad	Integer
Para cada responsabilidad:	
Nombre:	Get id_persona
Descripción:	Función que devuelve el identificador de una persona determinada.
Nombre:	Get nombre
Descripción:	Función que devuelve el nombre de una persona determinada.
Nombre:	Get 1er_Apellido
Descripción:	Función que devuelve el primer apellido de una persona determinada.
Nombre:	Get 2do_Apellido
Descripción:	Función que devuelve el segundo apellido de una persona determinada.
Nombre:	Get CI
Descripción:	Función que devuelve el número del carnet de identidad de una persona determinada.
Nombre:	Get fecha_nacimiento
Descripción:	Función que devuelve la fecha de nacimiento de una persona determinada.
Nombre:	Get nombre_madre
Descripción:	Función que devuelve el nombre de la madre de una persona determinada.
Nombre:	Get nombre_padre
Descripción:	Función que devuelve el nombre del padre de una persona determinada.
Nombre:	Get centro_trabajo
Descripción:	Función que devuelve el centro de trabajo al cual pertenece una persona

	determinada.
Nombre:	Get direccion_centro_trabajo
Descripción:	Función que devuelve la dirección del centro de trabajo de una persona determinada.
Nombre:	Get fecha_registro
Descripción:	Función que devuelve la fecha de registro de una persona determinada.
Nombre:	Get factor_rh
Descripción:	Función que devuelve el factor (rh) de una persona determinada.
Nombre:	Get telefono
Descripción:	Función que devuelve el teléfono de una persona determinada.
Nombre:	Get trabajador_salud
Descripción:	Función que devuelve si es o no trabajador de salud una persona determinada.
Nombre:	Get id_color_piel
Descripción:	Función que devuelve el identificador del color de piel de una persona determinada.
Nombre:	Get id_sexo
Descripción:	Función que devuelve el identificador del sexo de una persona determinada.
Nombre:	Get id_municipio
Descripción:	Función que devuelve el identificador del municipio al que pertenece una persona determinada.
Nombre:	Get id_provincia
Descripción:	Función que devuelve el identificador de la provincia a la que pertenece una persona determinada.
Nombre:	Get id_nacionalidad
Descripción:	Función que devuelve el identificador de la nacionalidad que tiene una persona determinada.
Nombre:	Set id_persona
Descripción:	Función que permite entrar el identificador de una persona determinada.
Nombre:	Set nombre

Descripción:	Función que permite entrar el nombre de una persona determinada.
Nombre:	Set 1er_Apellido
Descripción:	Función que permite entrar el primer apellido de una persona determinada.
Nombre:	Set 2do_Apellido
Descripción:	Función que permite entrar el segundo apellido de una persona determinada.
Nombre:	Set CI
Descripción:	Función que permite entrar el número del carnet de identidad de una persona determinada.
Nombre:	Set fecha_nacimiento
Descripción:	Función que permite entrar la fecha de nacimiento de una persona determinada.
Nombre:	Set nombre_madre
Descripción:	Permite entrar el nombre de la madre de una persona determinada.
Nombre:	Set nombre_padre
Descripción:	Función que permite entrar el nombre del padre de una persona determinada.
Nombre:	Set centro_trabajo
Descripción:	Función que permite entrar el nombre del centro de trabajo al cual pertenece una persona determinada.
Nombre:	Set direccion_centro_trabajo
Descripción:	Función que permite entrar la dirección del centro de trabajo de una persona determinada.
Nombre:	Set fecha_registro
Descripción:	Función que permite entrar la fecha de registro de una persona determinada.
Nombre:	Set factor_rh
Descripción:	Función que permite entrar el factor (rh) de una persona determinada.
Nombre:	Set telefono
Descripción:	Función que permite entrar el número de teléfono de una persona determinada.
Nombre:	Set trabajador_salud
Descripción:	Función que permite entrar si es o no trabajador de salud una persona determinada.

Nombre:	Set id_color_piel
Descripción:	Función que permite entrar el identificador del color de piel de una persona determinada.
Nombre:	Set id_sexo
Descripción:	Función que permite entrar el identificador del sexo de una persona determinada.
Nombre:	Set id_municipio
Descripción:	Función que permite entrar el identificador del municipio al que pertenece una persona determinada.
Nombre:	Set id_provincia
Descripción:	Función que permite entrar el identificador de la provincia a la que pertenece una persona determinada.
Nombre:	Set id_nacionalidad
Descripción:	Función que permite entrar el identificador de la nacionalidad que tiene una persona determinada.

TABLA 5. Descripción de la CE <CE_HC>

Nombre: CE_HC	
Tipo de clase: entidad	
Atributo	**Tipo**
id_persona	Integer
num_hc	Integer
fecha_inscripcion	Date
estado_fisico	String
en_archivo	Boolean
fecha_ultima_salida	Date

id_estado	Integer
Para cada responsabilidad:	
Nombre:	Get id_persona
Descripción:	Función que devuelve el identificador de una persona determinada.
Nombre:	Get num_hc
Descripción:	Función que devuelve el número de una HC determinada.
Nombre:	Get fecha_inscripcion
Descripción:	Función que devuelve la fecha de inscripción de una persona determinada.
Nombre:	Get estado_fisico
Descripción:	Función que devuelve el estado físico de una HC determinada.
Nombre:	Get en_archivo
Descripción:	Función que devuelve si se encuentra o no la HC en el archivo.
Nombre:	Get fecha_ultima_salida
Descripción:	Función que devuelve la fecha de la última salida de una HC determinada del archivo.
Nombre:	Get id_estado
Descripción:	Función que devuelve el identificador del estado de una HC determinada.
Nombre:	Set id_persona
Descripción:	Función que permite entrar el identificador de una persona determinada.
Nombre:	Set num_hc
Descripción:	Función que permite entrar el número de una HC determinada.
Nombre:	Set fecha_inscripcion
Descripción:	Función que permite entrar la fecha de inscripción de una persona determinada.
Nombre:	Set estado_fisico
Descripción:	Función que permite entrar el estado físico de una HC determinada.
Nombre:	Set en_archivo
Descripción:	Función que permite entrar si se encuentra o no la HC en el archivo.
Nombre:	Set fecha_ultima_salida

Descripción:	Función que permite entrar la fecha de la última salida de una HC determinada del archivo.
Nombre:	Set id_estado
Descripción:	Función que permite entrar el identificador del estado de una HC determinada.

TABLA 6. Descripción de la CE <CE_hc_estados>

Nombre: CE_EstadoHC	
Tipo de clase: entidad	
Atributo	**Tipo**
id_estado	Integer
estado	String
Para cada responsabilidad:	
Nombre:	Get id_estado
Descripción:	Función que devuelve el identificador del estado de una HC determinada.
Nombre:	Get estado
Descripción:	Función que devuelve el estado en que se encuentra una HC determinada.
Nombre:	Set id_estads
Descripción:	Función que permite entrar el identificador del estado de una HC determinada.
Nombre:	Set estado
Descripción:	Función que permite entrar el estado en que se encuentra una HC determinada.

TABLA 7. Descripción de la CE <CE_hc_union_historia_clinica>

Nombre: CE_hc_union_historia_clinica
Tipo de clase: entidad

Atributo	**Tipo**
hc_nueva	Integer
fecha_cambio	Date
login_union	Boolean
id_persona	Integer
Para cada responsabilidad:	
Nombre:	Get hc_nueva
Descripción:	Función que devuelve el número de la HC nueva que se quiere unir a la ya existente de un paciente determinado.
Nombre:	Get fecha_cambio
Descripción:	Función que devuelve la fecha en que fue realizada la unión de las HC de una persona determinada.
Nombre:	Get id_persona
Descripción:	Función que devuelve el identificador de una persona determinada.
Nombre:	Set hc_nueva
Descripción:	Función que permite entrar el número de la HC nueva que se quiere unir a la ya existente de un paciente determinado.
Nombre:	Set fecha_cambio
Descripción:	Función que permite entrar la fecha en que fue realizada la unión de las HC de una persona determinada.
Nombre:	Set id_persona
Descripción:	Función que permite entrar el identificador de una persona determinada.

TABLA 8. Descripción de la CE <CE_Ubicacion>

Nombre: CE_Ubicacion
Tipo de clase: entidad

Atributo	**Tipo**
piso	Integer
estante	String
no_gaveta	Integer
id_persona	Integer
num_hc	Integer

Para cada responsabilidad:	
Nombre:	Get piso
Descripción:	Función que devuelve el piso en el que se encuentra localizada una HC determinada.
Nombre:	Get estante
Descripción:	Función que devuelve el estante en el que se encuentra localizada una HC determinada.
Nombre:	Get no_gaveta
Descripción:	Función que devuelve el número de la gaveta en la que se encuentra localizada una HC determinada.
Nombre:	Get id_persona
Descripción:	Función que devuelve el identificador de una persona determinada.
Nombre:	Get num_hc
Descripción:	Función que devuelve el número de una HC determinada.
Nombre:	Set piso
Descripción:	Función que permite entrar el piso en el que se encuentra localizada una HC determinada.
Nombre:	Set estante
Descripción:	Función que permite entrar el estante en el que se encuentra localizada una HC determinada.
Nombre:	Set no_gaveta
Descripción:	Función que permite entrar el número de la gaveta en la que se encuentra localizada una HC determinada.
Nombre:	Set id_persona

Descripción:	Función que permite entrar el identificador de una persona determinada.
Nombre:	Set num_hc
Descripción:	Función que permite entrar el número de una HC determinada.

TABLA 9. Descripción de la CE <CE_RegE/S >

Nombre: CE_RegE/S	
Tipo de clase: entidad	
Atributo	**Tipo**
id_registro	Integer
fecha_hora_entrega	Date
fecha_hora_retorno	Date
entregada_a	String
recibida_de	String
recibida_por	String
Para cada responsabilidad:	
Nombre:	Get id _ registro
Descripción:	Función que devuelve el identificador del registro de entrada o salida de una HC determinada.
Nombre:	Get fecha_hora_entrega
Descripción:	Función que devuelve la fecha y la hora en que salió del archivo una HC determinada.
Nombre:	Get fecha_hora_retorno
Descripción:	Función que devuelve la fecha y la hora en que fue devuelta una HC determinada.
Nombre:	Get entregada_a
Descripción:	Función que devuelve el nombre al que fue entregada una HC determinada.
Nombre:	Get recibida_de
Descripción:	Función que devuelve el nombre del que realizó la entrega de una HC determinada.

Nombre:	Get recibida_por
Descripción:	Función que devuelve el nombre del que recibió de vuelta una HC determinada.
Nombre:	Set id _ registro
Descripción:	Función que permite entrar el identificador del registro de entrada o salida de una HC determinada.
Nombre:	Set fecha_hora_entrega
Descripción:	Función que permite entrar la fecha y la hora en que salió del archivo una HC determinada.
Nombre:	Set fecha_hora_retorno
Descripción:	Función que permite entrar la fecha y la hora en que fue devuelta una HC determinada.
Nombre:	Set entregada_a
Descripción:	Función que permite entrar el nombre al que fue entregada una HC determinada.
Nombre:	Set recibida_de
Descripción:	Función que permite entrar el nombre del que realizó la entrega de una HC determinada.
Nombre:	Set recibida_por
Descripción:	Función que permite entrar el nombre del que recibió de vuelta una HC determinada.

TABLA 10. Descripción de la CE <CE_Departamento>

Nombre: CE_Departamento	
Tipo de clase: entidad	
Atributo	**Tipo**
id_departamento	Integer
departamento	String
Para cada responsabilidad:	
Nombre:	Get id_departamento

Descripción:	Función que devuelve el identificador de un departamento determinado.
Nombre:	Get departamento
Descripción:	Función que devuelve el nombre de un departamento determinado.
Nombre:	Set id_departamento
Descripción:	Función que permite entrar el identificador de un departamento determinado.
Nombre:	Set departamento
Descripción:	Función que permite entrar el nombre de un departamento determinado.

TABLA 11. Descripción de la CE <CE_Motivo_Salida>

Nombre: CE_Motivo_Salida	
Tipo de clase: entidad	
Atributo	**Tipo**
id_ motivo_salida	Integer
motivo_salida	String
Para cada responsabilidad:	
Nombre:	Get id_ motivo_salida
Descripción:	Función que devuelve el identificador del motivo de salida por el cual fue movida del archivo una HC determinada.
Nombre:	Get motivo_salida
Descripción:	Función que devuelve el motivo de salida de una HC determinada del archivo.
Nombre:	Set id_ motivo_salida
Descripción:	Función que permite entrar el identificador del motivo de salida por el cual fue movida del archivo una HC determinada.
Nombre:	Set motivo_salida
Descripción:	Función que permite entrar el motivo de salida de una HC determinada del archivo.

II.III.II. CLASES CONTROLADORAS (CC).

TABLA 12. Descripción de la CC <Autenticar Usuario>

Nombre: CC_VerificarUsuario	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	VerificarUsuario
Descripción:	Función que permite verificar los datos del usuario.
Nombre:	InsertarUsuario
Descripción:	Función que permite insertar un nuevo usuario.
Nombre:	ModificarRol
Descripción:	Función que permite modificar el rol que tiene el usuario
Nombre:	EliminarUsuario
Descripción:	Función que permite eliminar los datos de un usuario.

TABLA 13. Descripción de la CC del repositorio de la CC <Autenticar Usuario>

Nombre: CC_RepositorioAutenticar	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	SeleccionarUsuario
Descripción:	Función que permite definir la selección de los datos del usuario.
Nombre:	InsertarUsuario
Descripción:	Función que permite definir los datos del un nuevo usuario ha insertar.

Nombre:	ModificarRol
Descripción:	Función que permite definir el rol del usuario ha modificar.
Nombre:	EliminarUsuario
Descripción:	Función que permite definir los datos de un usuario que serán eliminados

TABLA 14. Descripción de la CC <Gestionar HC>

Nombre: CC_Gestionar HC	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	BuscarPersona
Descripción:	Función que define la búsqueda de la HC del paciente por los datos personales de este.
Nombre:	ModificarEstadoHC
Descripción:	Función que define la unión de una HC provisional con la HC del paciente.
Nombre:	ModificarUbicacion
Descripción:	Función que define el cambio de la ubicación de las HC.
Nombre:	VerificarUbicacion
Descripción:	Función que define la verificación de la ubicación de las HC.
Nombre:	InsertarHC
Descripción:	Función que define que los datos de las HC sean insertados
Nombre:	InsertarUbicación
Descripción:	Función que define que los datos de la ubicación de la HC sean insertados.

Nombre:	SeleccionarUbicación
Descripción:	Función que define que los datos para seleccionar la ubicación de la HC
Nombre:	BuscarHC
Descripción:	Función que define la búsqueda de las HC.
Nombre:	BuscarEstado
Descripción:	Función que define la búsqueda del estado de las HC
Nombre:	UnionHC
Descripción:	Función que define el cambio del estado de la HC.
Nombre:	InsertarNumHCSecundaria
Descripción:	Función que define que los datos de la HC provisional sean insertados.
Nombre:	RegistroEntrada
Descripción:	Función que define el registro de los datos de entrada de la HC al archivo.
Nombre:	RegistroSalida
Descripción:	Función que define el registro de los datos de salida de la HC al archivo.
Nombre:	BuscarDepartamento
Descripción:	Función que define la búsqueda del departamento que solicitó las HC.
Nombre:	BuscarMotSalida
Descripción:	Función que define la búsqueda del motivo de salida con que salieron las HC del archivo.

TABLA 15. Descripción de la CC del repositorio de la CC <Gestionar HC>

Nombre: CC_Repositorio_Gestionar_HC	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	

Nombre:	InsertarHC
Descripción:	Función que permite definir los datos de las HC que van hacer insertados
Nombre:	InsertarNumHC
Descripción:	Función que permite definir que el número de la HC sea insertado.
Nombre:	InsertarUbicacion
Descripción:	Función que permite definir los datos de la ubicación de las HC ha insertar.
Nombre:	InsertarSalida
Descripción:	Función que permite definir los datos de salida de la HC ha insertar.
Nombre:	InsertarEntrada
Descripción:	Función que permite definir los datos de entrada de la HC ha insertar.
Nombre:	ActualizarUnion
Descripción:	Función que permite definir la actualización de los datos de la unión realizada con la HC provisional y la HC del paciente.
Nombre:	ActualizarEstadoHC
Descripción:	Función que permite definir la actualización del estado de la HC.
Nombre:	SeleccionarPersona
Descripción:	Función que permite definir la selección de pacientes.
Nombre:	SeleccionarEstado
Descripción:	Función que permite definir la selección del estado de la HC.
Nombre:	SeleccionarUbicacion
Descripción:	Función que permite definir la selección de la ubicación de la HC.
Nombre:	SeleccionarDepartamento
Descripción:	Función que permite definir la selección del departamento que solicitó la HC.
Nombre:	SeleccionarMotSalida
Descripción:	Función que permite definir la selección del motivo de salida con que salió la HC del archivo.

TABLA 16. Descripción de la CC <Buscar HC>

Nombre: CC_Buscar HC	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	BuscarHCNum
Descripción:	Función que define la búsqueda por el número de la HC.
Nombre:	BuscarPersona
Descripción:	Función que define la búsqueda por los datos del paciente.
Nombre:	BuscarEstadoHC
Descripción:	Función que define la búsqueda por el estado de la HC.
Nombre:	BuscarUbicacion
Descripción:	Función que define la búsqueda por la ubicación de la HC.
Nombre:	BuscarRegistroE/S
Descripción:	Función que define la búsqueda por el registro de entrada/salida en que este la HC.
Nombre:	MostrarTodosDatosHC
Descripción:	Función que define la presentación de todos los datos de la HC.
Nombre:	BuscarMotivo_Salida
Descripción:	Función que define la búsqueda por el motivo de salida con que hayan salido las HC.
Nombre:	BuscarHCDepartamentos
Descripción:	Función que define la búsqueda por los departamentos que hayan solicitado/entregado las HC.
Nombre:	BuscarHC_EstadoFisico
Descripción:	Función que define la búsqueda por el estado físico que tenga la HC.

TABLA 17. Descripción de la CC del repositorio de la CC <Buscar HC>

Nombre: CC_Repositorio_Buscar HC	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	SeleccionarPersona
Descripción:	Función que define la selección del paciente por sus datos.
Nombre:	SeleccionarEstado
Descripción:	Función que define la selección de la HC por el estado.
Nombre:	SeleccionarUbicacion
Descripción:	Función que define la selección de la ubicación de la HC.
Nombre:	SeleccionarRegistroE/S
Descripción:	Función que define la seleción del registro de entrada/salida en que este la HC.
Nombre:	SeleccionarDepartamento
Descripción:	Función que define la selección del departamento que solicito la HC.
Nombre:	SeleccionarMotivo_Salida
Descripción:	Función que define la selección del motivo de salida con que hayan salido las HC.

TABLA 18. Descripción de la CC <Crear Reporte>

Nombre: CC_Crear Reporte	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	

Nombre:	CrearReporte
Descripción:	Función que define la presentación del reporte creado

TABLA 19. Descripción de la CC del repositorio de la CC <Crear Reporte>

Nombre: CC_Repositorio_Crear Reporte	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	SeleccionarEstado
Descripción:	Función que permite definir la selección del estado de la HC.
Nombre:	SeleccionarUbicacion
Descripción:	Función que permite definir la selección de la ubicación de la HC.
Nombre:	SeleccionarDepartamento
Descripción:	Función que permite definir la selección del departamento que solicitó la HC.
Nombre:	SeleccionarMotSalida
Descripción:	Función que permite definir la selección del motivo de salida con que salió la HC del archivo.
Nombre:	SeleccionarHC
Descripción:	Función que permite definir la selección de la HC.
Nombre:	SeleccionarPersona
Descripción:	Función que permite definir la selección del paciente.
Nombre:	SeleccionarRegE/S
Descripción:	Función que permite definir la selección del registro de entrada/salida de la HC del archivo.

II.III.III. CLASES INTERFACES (CI).

TABLA 20. Descripción de la CI del CUS <Autenticar Usuario>

Nombre: CI_AutenticarUsuario	
Tipo de clase: interfaz	
Atributo	**Tipo**
txt_Usuario	textbox
txt_Contraseña	textbox
btnAceptar	button
btnCancelar	button
Para cada responsabilidad:	
Nombre:	protected void btnAceptar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnCancelar_Click
Descripción:	Función que define la programación del evento de un botón.

TABLA 21. Descripción de la CI Eliminar Usuario del CUS <Autenticar Usuario>

Nombre: CI_EliminarUsuario	
Tipo de clase: interfaz	
Atributo	**Tipo**
txt_Num_Identidad	textbox
txt_Usuario	textbox
txt_Rol	textbox
btnCancelar	button
btnEliminarUsuario	button

Para cada responsabilidad:	
Nombre:	protected void btnCancelar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnEliminar_Click
Descripción:	Función que define la programación del evento de un botón.

TABLA 22. Descripción de la CI Insertar Usuario del CUS <Autenticar Usuario>

Nombre: CI_InsertarUsuario	
Tipo de clase: interfaz	
Atributo	**Tipo**
txt_Nombre_Usuario	textbox
txt_Contraseña	textbox
txt_Confirmar_Contraseña	textbox
txt_Num_Identidad	textbox
txt_Rol	textbox
btnCancelar	button
btnAdicionar	button
Para cada responsabilidad:	
Nombre:	protected void btnCancelar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnAdicionar_Click
Descripción:	Función que define la programación del evento de un botón.

TABLA 23. Descripción de la CI Modificar Rol del CUS <Autenticar Usuario>

Nombre: CI_ModificarRol	
Tipo de clase: interfaz	
Atributo	**Tipo**
txt_Nombre_Usuario	textbox
txt_Num_Identidad	textbox
txt_Rol	textbox
txt_Permisos	textbox
btnCancelar	button
btnModificar	button
Para cada responsabilidad:	
Nombre:	protected void btnCancelar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnAdicionar_Click
Descripción:	Función que define la programación del evento de un botón.

TABLA 24. Descripción de la CI del CUS <Clasificar HC>

Nombre: CI_Clasificar HC	
Tipo de clase: interfaz	
Atributo	**Tipo**
txt_num_HC	textbox
txt_nombre	textbox
txt_CI	textbox
cbx_estado	combobox
txt_ubicacion	textbox
btnBuscar	button
btnAceptar	button

btnCancelar	button
Para cada responsabilidad:	
Nombre:	protected void btnBuscar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnAceptar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnCancelar_Click
Descripción:	Función que define la programación del evento de un botón.

TABLA 25. Descripción de la CI del CUS <Insertar HC>

Nombre: CI_Insertar HC	
Tipo de clase: interfaz	
Atributo	**Tipo**
txt_num_HC	textbox
txt_nombre	textbox
txt_1er_apellido	textbox
txt_2do_apellido	textbox
txt_CI	textbox
cbx_estado	combobox
txt_ubicacion	textbox
btnBuscar	button
btnAceptar	button
btnCancelar	button
Para cada responsabilidad:	
Nombre:	protected void btnAceptar_Click
Descripción:	Función que define la programación del evento de un botón.

Nombre:	protected void btnCancelar_Click
Descripción:	Función que define la programación del evento de un botón.

TABLA 26. Descripción de la CI del CUS <Unir HC>

Nombre: CI_Unir HC	
Tipo de clase: interfaz	
Atributo	**Tipo**
txt_num_HC	textbox
txt_direccion	textbox
txt_edad	textbox
txt_ubicacion	textbox
btnAceptar	button
btnCancelar	button
Para cada responsabilidad:	
Nombre:	protected void btnBuscar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnAceptar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnCancelar_Click
Descripción:	Función que define la programación del evento de un botón.

TABLA 27. Descripción de la CI del CUS <Registrar E/S HC>

Nombre: CI_Registrar E/S HC
Tipo de clase: interfaz

Atributo	**Tipo**
date_fecha_hora_entrega	date
txt_prestado_a	textbox
date_fecha_hora_retorno	date
txt_entregada_por	textbox
txt_observaciones	textbox
cbx_motivo_salida	combobox
cbx_departamento	combobox
btnAceptar	button
btnCancelar	button
Para cada responsabilidad:	
Nombre:	protected void btnAceptar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnCancelar_Click
Descripción:	Función que define la programación del evento de un botón.

TABLA 28. Descripción de la CI del CUS <Buscar HC>

Nombre: CI_Buscar HC	
Tipo de clase: interfaz	
Atributo	**Tipo**
txt_nombre	textbox
txt_1er_apellido	textbox
txt_2do_apellido	textbox
txt_CI	textbox
txt_num_HC	textbox
date_fecha_nacimiento	date

cbx_sexo	combobox
cbx_estado	combobox
txt_ubicacion	textbox
date_fecha_ entrada	date
date_fecha_ salida	date
cbx_motivo_salida	combobox
txt_en_archivo	textbox
btnAceptar	button
btnCancelar	button
Para cada responsabilidad:	
Nombre:	protected void btnAceptar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnCancelar_Click
Descripción:	Función que define la programación del evento de un botón.

TABLA 29. Descripción de la CI del CUS <Crear Reporte>

Nombre: CI_Crear Reporte	
Tipo de clase: interfaz	
Atributo	**Tipo**
dtp_desde	datetimepicker
dtp_hasta	datetimepicker
chk_motivo	checkbox
chk_fechaE/S	checkbox
chk_estadofisicoHC	checkbox
chk_departamento	checkbox
chk_estadoHC	checkbox

btnAceptar	button
btnCancelar	button
Para cada responsabilidad:	
Nombre:	protected void btnAceptar_Click
Descripción:	Función que define la programación del evento de un botón.
Nombre:	protected void btnCancelar_Click
Descripción:	Función que define la programación del evento de un botón.

En este capítulo se realiza una valoración crítica del diseño propuesto por los analistas, teniendo en cuenta la correcta definición de las clases fundamentales del sistema a construir, además de destacarse los patrones de diseño utilizados para la confección y modelado del negocio. Se decidió la reutilización de la dll Npgsql, para la conexión con la base de datos relacionada con la aplicación; así como también, la arquitectura propuesta para algunos de los módulos pertenecientes al Sistema de Información Hospitalaria en construcción, como son: Seguridad, Configuración e Inscripción-Admisión. Se realizó una detallada descripción de las clases utilizadas, dejándose ver claramente los métodos y atributos correspondientes a cada una de ellas.

Capitulo III. Validación de la solución propuesta.

Durante el proceso de desarrollo de un software, los errores pueden comenzar a aparecer incluso desde el mismo momento en que fueron definidos los objetivos y estos a su vez especificados de forma errónea e imperfecta; de la misma forma en los posteriores pasos del diseño y desarrollo. A raíz de la incapacidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software debe ser acompañado de una actividad que garantice su calidad.

La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

La creciente inclusión del software, como un elemento más de muchos sistemas y la importancia de los costos asociados a un fallo del mismo, han motivado la creación de pruebas más minuciosas y bien planificadas.

III.I. Pruebas aplicadas.

Las pruebas son el proceso de ejercitar un programa con la intención específica de encontrar errores previos a la entrega al usuario final.

Las pruebas son la actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, donde los resultados son observados y registrados, y es hecha una evaluación de algún aspecto del sistema o componente.

Con la realización de estas pruebas se pretende encontrar y documentar los defectos que puedan afectar la calidad del software, validar y probar los requisitos que debe cumplir el software y a su vez que estos fueron implementados correctamente.

Es necesario analizar que las pruebas no pueden asegurar la ausencia de defectos sino que permiten demostrar que existen defectos en el software, que cada prototipo que se quiera entregar al final de una iteración debe ser probado y evaluado.

Los casos de prueba especifican la forma de probar el sistema en cuestión, incluyendo la entrada o resultado con el que se ha de probar y las condiciones bajo las que ha de probarse. Es un conjunto de entradas y resultados esperados que ponen en práctica el funcionamiento de un componente con el objetivo de causar fallas y detectar defectos. Entre los casos de prueba pueden existir todos los tipos de relaciones, pero las más importantes son las de precedencia.

Todo producto puede ser probado de las siguientes formas:

1) Conociendo el funcionamiento del producto, para poder desarrollar pruebas que aseguren que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.
2) Conociendo la funcionalidad específica para la cual fue diseñado el producto, para poder llevar a cabo pruebas que demuestren que cada función es completamente operativa.

En el primer caso, las pruebas están dirigidas a la aplicación de métodos basados en Caja Blanca y en el segundo caso, a los métodos basados en Caja Negra.

Los de este primer grupo permiten la comprobación de los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones.

Es recomendable que a la hora de plantearse una estrategia de pruebas para comprobar la calidad del código de la aplicación se tengan en cuenta a los del primer grupo, en el caso particular de esta aplicación se realizó una prueba a nivel de unidad empleándose la técnica del Camino Básico. Esta técnica permite obtener una medida de la complejidad lógica del diseño procedimental usándola como guía para la definición de un conjunto básico de caminos de ejecución.

¿Cuáles son los pasos que se siguen para aplicar esta técnica?

Los pasos que se siguen para aplicar esta técnica son:

1) A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2) Se calcula la complejidad ciclomática del grafo.

3) Se determina un conjunto básico de caminos independientes.
4) Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

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

Existen tres formas fundamentales de calcular la complejidad:

- El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
- La complejidad ciclomática, V(G), se define como:

$$V(G) = A - N + 2$$

donde: A es el número de aristas del grafo y N es el número de nodos.

- La complejidad ciclomática, V(G), también se define como:

$$V(G) = P + 1$$

donde: P es el número de nodos predicado contenidos en el grafo G.

¿Cómo se obtienen los casos de prueba?

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

Luego de confeccionar los casos de prueba se ejecutan cada uno de estos y se comparan los resultados con los esperados. Una vez terminados todos los casos de prueba, se estará seguro de que todas las sentencias del programa se han ejecutado por lo menos una vez.

Es importante considerar que algunos caminos no se pueden probar de forma aislada, o sea, la combinación de datos requeridos para recorrer el camino no se puede obtener con el flujo normal del programa. En tales casos, estos caminos se prueban como parte de otra prueba de camino.

Entrando en el segundo grupo de métodos de prueba, se puede enfatizar que la prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

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

Como técnica de este grupo tenida en cuenta a la hora de comprobar la calidad del producto está la técnica de la Partición de Equivalencia: que divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Dentro del método de Caja Negra la técnica de la *Partición de Equivalencia* es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de prueba que hay que desarrollar.

III.I.I. Descripción de los casos de pruebas de Caja Blanca aplicados.

Para llevar a cabo los casos de pruebas de caja blanca previstas en el epígrafe anterior, fue preciso tomar muestra del código fuente de la aplicación, para esto se tuvo en cuenta la implementación del método ValidarSalida.

En este caso se determinó el grafo de flujo y por ende la complejidad ciclomática, obteniéndose la cantidad de caminos independientes necesarios para aplicar los casos de pruebas correspondientes a cada cual.

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

III.I.I.I. CASOS DE PRUEBAS.

a. Siguiendo los pasos de la Técnica del Camino Básico se obtuvo por cada clase:

```
public class CtrlControlEntradaSalida
  {
    private bool ValidarSalida(string pNumeroHC)
    {
      HistoriaClinica hc = new HistoriaClinica();  1
      hc.NumeroHc = pNumeroHC;  1
      HistoriaClinicaRepositorio hcr = new HistoriaClinicaRepositorio();  1
      HistoriaClinica hc2 =new HistoriaClinica();  1
      hc2=hcr.ObtenerUno(hc);  1
      if (hc2.FechaInscripcion == null)  2
        throw new Exception("La HC no está registrada");  3
      if (hc.EnArchivo == false)  4
        throw new Exception("La HC ya ha salido de Archivo");  5
      return true;  6
    }  7
```

Fig.1 Método ValidarSalida

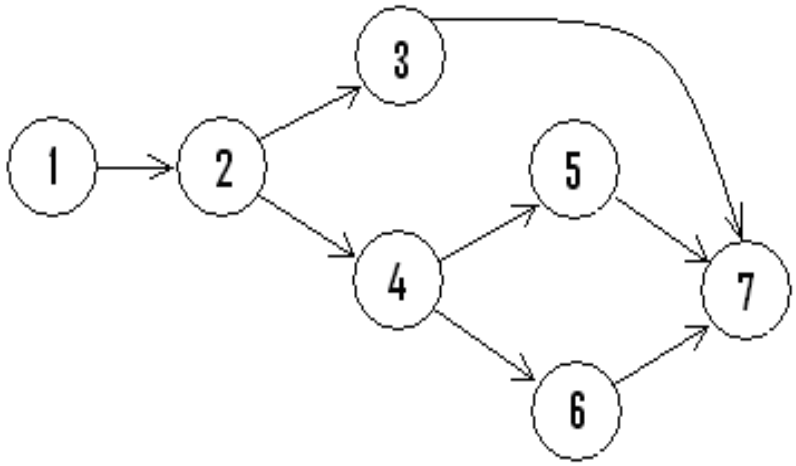


Fig.2 Representación del Grafo de flujo del Método ValidarSalida.

En este primer caso se obtuvo que:

- el grafo de flujo tiene tres regiones.
- V (G) = 8 aristas - 7 nodos + 2 = 3.
- V (G) = 2 nodos predicados + 1 = 3.

Por tanto la complejidad ciclomática del grafo de flujo de la Figura 2 es igual a 3.

Como caminos independientes se determinaron:

Camino1: 1-2-3-7

Camino2: 1-2-4-5-7

Camino3: 1-2-4-6-7

Para estos caminos independientes se conformaron los siguientes casos de pruebas:

Caso de prueba para el Camino1:

Descripción y asignación:

Se quiere determinar si dado un número de HC, esta se encuentra o no fuera del archivo, para esto es asignado a la variable pNumeroHC un número de HC que no existe registrado en la base de datos debido a que no existe una HC con tal número y por lo tanto no hay una fecha de inscripción o registro almacenada en la base de datos, siguiendo estas especificaciones se garantiza que se recorra este camino independiente en su totalidad.

Resultado esperado:

Al entrar el número y comprobar que no existe una fecha de inscripción perteneciente a la HC correspondiente a este número, el sistema muestra el mensaje "La HC no está registrada", avisando que no está registrada en archivo esta HC.

Caso de prueba para el Camino 2:

Descripción y asignación:

Se quiere determinar si dado un número de HC, esta se encuentra o no fuera del archivo, para ello es asignado a la variable pNumeroHC un número de HC que corresponde al de una de las historias clínicas registradas en la base de datos, existiendo así una fecha de inscripción o registro correspondiente a la misma, pero esta HC ha sido movida del archivo por algún motivo, siguiendo estas especificaciones se garantiza que se recorra este camino independiente en su totalidad.

Resultado esperado:

Al entrar el número y comprobar que existe una fecha de inscripción perteneciente a la HC correspondiente a este número, pero que ha sido movida del archivo, el sistema muestra el mensaje "La HC ya ha salido de Archivo", avisando así, que no se encuentra en archivo esta HC.

Caso de prueba para el Camino 3:

Descripción y asignación:

Se quiere determinar si dado un número de HC, esta se encuentra o no fuera del archivo, para ello es asignado a la variable pNumeroHC un número de HC que corresponde al de una de las historias clínicas registradas en la base de datos, existiendo así una fecha de inscripción o registro correspondiente a la misma, pero esta HC no ha sido movida del archivo, o sea, permanece en el mismo, siguiendo estas especificaciones se garantiza que se recorra este camino independiente en su totalidad.

Resultado esperado:

Al entrar el número y comprobar que existe una fecha de inscripción perteneciente a la HC correspondiente a este número y que se mantiene dentro del archivo, el sistema devuelve el valor de verdadero, que se traduce en que sí se encuentra aún en el archivo la HC solicitada.

b. Evaluación de los resultados obtenidos.

Los resultados de la realización de los casos de pruebas de caja blanca para el caso de Validar la salida de las historias clínicas del archivo, fueron satisfactorios al lograr determinar los movimientos realizados con una HC determinada, teniendo en cuenta la importancia que tiene el control de este documento médico que contiene toda la información del paciente y que es de tan vital importancia para el tratamiento y seguimiento del mismo.

III.I.II. Descripción de los casos de pruebas de Caja Negra aplicados.

Inicialmente, fue escogida una de las interfaces que componen la aplicación y se determinaron las clases correspondientes a tener en cuenta durante la realización del proceso de pruebas. La interface en analizada fue la correspondiente a la funcionalidad “Unir HC”.

III.I.II.I CASOS DE PRUEBAS.

a. Siguiendo los pasos de la Técnica de la Partición Equivalente se analizó lo siguiente:

Propósito:

Probar el correcto funcionamiento de la funcionalidad “Unir HC”.

Descripción General del Caso de Uso:

Este proceso se ejecuta con el fin de introducir los datos necesarios para la unión de dos historias clínicas, quedando como resultado el resumen en una sola con todos los datos representados en las historias anteriores, invitándose así, la presencia de dos informes médicos correspondientes a un mismo paciente.

Flujo Central:

- El sistema muestra la interfaz principal de la aplicación y el usuario selecciona en el menú Historia Clínica la opción: “Unir HC”.

  - El sistema muestra la interfaz Unir HC, donde el usuario puede introducir el número de la HC existente en el archivo y luego de seleccionar el botón “Buscar”, obtiene la ubicación de la misma, entonces puede proceder a entrar los datos relacionados con la nueva HC y si lo desea, puede además, modificar la ubicación de la resultante HC dentro del archivo.

- Al concluir con este proceso de entrada de datos y el usuario haber seleccionado el botón “Aceptar” el sistema muestra el mensaje “Se ha realizado satisfactoriamente la unión de las HC” dándose de esta forma por terminado el proceso en cuestión.

Nombre: "Unir HC**"**			
Clases válidas	**Clases inválidas**	**Resultado esperado**	**Resultado de la prueba**
"#HC" = "84012321616" Presionar botón "Buscar"		Al presionar el botón "Buscar" aparecen los datos relacionados con el #HC entrado y se procede entonces a entrar los restantes datos correspondientes a la nueva HC, los cuales son guardados en un único informe.	Satisfactorio
	"#HC" = "84012321616" Presionar botón "Buscar"	El sistema muestra un mensaje de notificación indicando que el #HC no se encuentra registrado en el sistema.	Satisfactorio
	"#HC" = "abcdefghi" Presionar botón "Buscar"	Al presionar el botón "Buscar", el sistema muestra un mensaje de error especificando que la entrada del #HC no es válida (solo se admiten números), señalará en color rojo los campos requeridos que deben ser llenados correctamente.	Satisfactorio

b. Evaluación de los resultados obtenidos.

Los resultados de la realización de los casos de pruebas de Caja Negra para la interfaz referente a la funcionalidad "Unir HC", resultaron satisfactorios al poder comprobar que las funciones esperadas de esta

parte de la aplicación se ejecutaron correctamente, cumpliéndose de esta forma parte del objetivo propuesto en el trabajo.

Es preciso reflexionar y concluir que a cada defecto encontrado durante la realización de los casos de pruebas se le debe establecer una prioridad que determine la importancia y necesidad de rapidez a la hora de solucionar el error existente. Generalmente es práctico y suficiente establecer cuatro niveles de prioridad para los errores, ellos son:

1. Prioridad Crítica (Aquel error que necesita ser solucionado Inmediatamente).
2. Prioridad Alta.
3. Prioridad Media.
4. Prioridad Baja.

En el presente caso de prueba, no fue detectado ningún error a la hora desarrollar la técnica de la Partición de Equivalencia, obteniéndose un resultado satisfactorio con la aplicación de la misma.

El objetivo de la prueba de software es descubrir errores. Para conseguir este objetivo se planifica y se ejecuta una serie de pasos que van revisando todos los elementos del software.

Una vez concluido este capítulo se han desarrollado todas las fases del proyecto, probándose la calidad del software que se ha ido construyendo, aunque como el grueso de la programación se realiza en la construcción, es en esa fase en la que se centran los mayores esfuerzos de este flujo. Se realizó la etapa de pruebas, que es tan o más importante que todas las realizadas hasta el momento puesto que en ella se refleja la calidad con que ha sido llevada a cabo la proyección del sistema.

Para comprobar la optimización y eficiencia del código fuente generado durante la implementación de la aplicación, se utilizó la técnica del Camino Básico, basada en el método de prueba de Caja Blanca. Luego de haber alcanzado la interfaz visual de esta aplicación, para comprobar el cumplimiento de las funcionalidades requeridas por el sistema; se tuvo en cuenta la técnica de la Partición Equivalente, basada en el método de prueba de Caja Negra.

Conclusiones

Como resultado final de este trabajo de diploma, se logró implementar una aplicación de escritorio que permite llevar a cabo las funcionalidades necesarias para el control de las HC en los departamentos de archivo/HC en los centros médicos del país, brindando la adecuada seguridad para el almacenamiento en la base de datos correspondiente al sistema implementado, desde los puntos de vista de ataque de intrusos al sistema y de pérdida de datos a la hora de operar con el sistema por parte del personal autorizado.

Recomendaciones

Con este trabajo se exhorta a una profundización del tema abordado para ampliar los conocimientos con vista a detectar posibles debilidades en la implementación de la aplicación.

- Confeccionar un sistema de ayuda que permita comprender de forma fácil y entendible el manejo de la presente aplicación.
- Lograr en un futuro la implementación de una aplicación o sistema Web que brinde las funcionalidades existentes y las que podrían aparecer a lo largo de la prueba piloto de la presente versión realizada como aplicación de escritorio.

Referencias bibliográficas

1. (CIM_UAG), C. D. I. M. *HIS 4.0*. Disponible en: http://cim.uag.mx/his.html

2. MEDICAL-SOFT. *MedFile 5.x*. Disponible en: http://www.medical-soft.com/

3. SIVS.A. *Hosix-V*, 2007. Disponible en: http://www.sivsa.com/

4. CARE2X. *CARE2X*, 2007. Disponible en: http://care2x.org/index.php?c2x_lang=es&chglang=1

5. C-DAC. *C-DAC'S Sushrut* 2006. Disponible en: http://www.cdac.in/html/his/sushrut.asp

6. SOBREVILLA, A. *Galenhos (R)*, Octubre 2006. Disponible en: http://www.elhospital.com/eh/secciones/EH/ES/MAIN/IN/ESTUDIOS_CASO/doc_52343_HTML.html?idDocumento=52343%20-%2051k%20-%202007-04-21

7. *Plataforma .Net*. Disponible en: http://es.wikipedia.org/wiki/.NET

8. *Mono*. Disponible en: http://www.pedromj.com/tecnologias/mono.aspx

9. *Lenguaje C#*. Disponible en: http://es.wikipedia.org/wiki/C_Sharp

10. *Patrones de diseño*. 2007. Disponible en: http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php

11. *Gestor de Base de Datos*. 2007. Disponible en: http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php

12. *Clases*. 2007. Disponible en: http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_1.htm

13. *Clases abstractas*. 2007. Disponible en:

http://www.fi-b.unam.mx/pp/profesores/carlos/java/java_basico4_8.html

14. *Compiladores*. 2007. Disponible en: http://www.dimelo.com/GS/SeeTheme.aspx?item=60

15. *Frames*. 2007. Disponible en: http://www.desarrolloweb.com/articulos/791.php

16. *Funciones*. 2007. Disponible en: http://www.desarrolloweb.com/manuales/20/

17. *Gestor de Base de Datos*. 2007. Disponible en: http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php

18. *Licencia LGPL*. 2007. Disponible en: http://creativecommons.org/licenses/LGPL/2.1/

19. *Orientado a objetos*. 2007. Disponible en: http://www.monografias.com/trabajos5/softobj/softobj.shtml

20. *Servidor*. 2007. Disponible en: http://www.mty.itesm.mx/rectoria/dda/usols/concepto1.htm

21. *Sistemas operativos*. 2007. Disponible en: http://www.mflor.mx/materias/comp/cursoso/sisope1.htm

22. *Data warehouse*. 2007. Disponible en: http://www.abcdatos.com/tutoriales/tutorial/l7781.html

23. *ECMA*. 2007. Disponible en: http://www.albertolacalle.com/diseno-estandares.htm

Bibliografía

(CIM_UAG), C. D. I. M. *HIS 4.0*. Disponible en: http://cim.uag.mx/his.html

Introducción a Oracle. 2006]. Disponible en: http://www.mailxmail.com/curso/informatica/introduccionoracle/toc.htm

SharpDevelop, alternativas libres. 2007]. Disponible en: http://alts.homelinux.net/libreapp.php?id=274

CECAM. *Sistema Automatizado de Egresos Hospitalarios y Cálculo de Indicadores Ajustados*. 2006]. Disponible en: http://www.cecam.sld.cu/pages/rcim/revista_6/articulo_htm/sistema.htm

ESLOMAS. *Introducción a la plataforma .NET y Mono*, 2007]. Disponible en: http://www.eslomas.com/index.php/archives/2005/05/11/introduccion-plataforma-net-y-mono/1/

FACTORY, D. O. *Design Patterns*, 2001. [2007]. Disponible en: http://www.dofactory.com/Patterns/Patterns.aspx

FDL, *G. Mono*, 2007]. Disponible en: http://www.mono-project.com/Main_Page

GUTIERREZ, J. A. S. *PATRONES GRASP (Patrones de Software para la asignación General de Responsabilidad).Parte II*, 2006. [2007]. Disponible en: http://jorgesaavedra.wordpress.com/tag/patrones-grasp/

Tabla de equivalencias entre VB.Net y CSharp (C#), 2006. [2007]. Disponible en: http://jorgesaavedra.wordpress.com/about/

LIVERPOOL, T. U. O. *Definición arquitectura cliente servidor*, 1997. [2006]. Disponible en: http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml

BIBLIOGRAFIA

MVP, C. *DESARROLLADOR PROFESIONAL DE APLICACIONES CON .NET* 2006]. Disponible en: http://www.campusmvp.com/CampusMVP/Cursos_On-Line.htm

SERVICES, H. C. I., 2005. [2006]. Disponible en: http://www.hcisonline.com/online/sp/software.asp?id=5

TARDAGUILA, C. Un ejemplo del patrón abstract factory, 2004. [2007]. Disponible en: http://www.design-nation.net/es/archivos/000410.php

WEBDEVSTUDIO. *IDE*, 2006]. Disponible en: http://webdevstudio.wordpress.com/descripcion/

Manual de mySQL. Disponible en: http://www.programatium.com/tutoriales/cursos/mysql/index.htm

Taller de MySQL. Disponible en: http://www.desarrolloweb.com/manuales/34/

Tutorial de SQL. Disponible en: http://www.desarrolloweb.com/manuales/9/

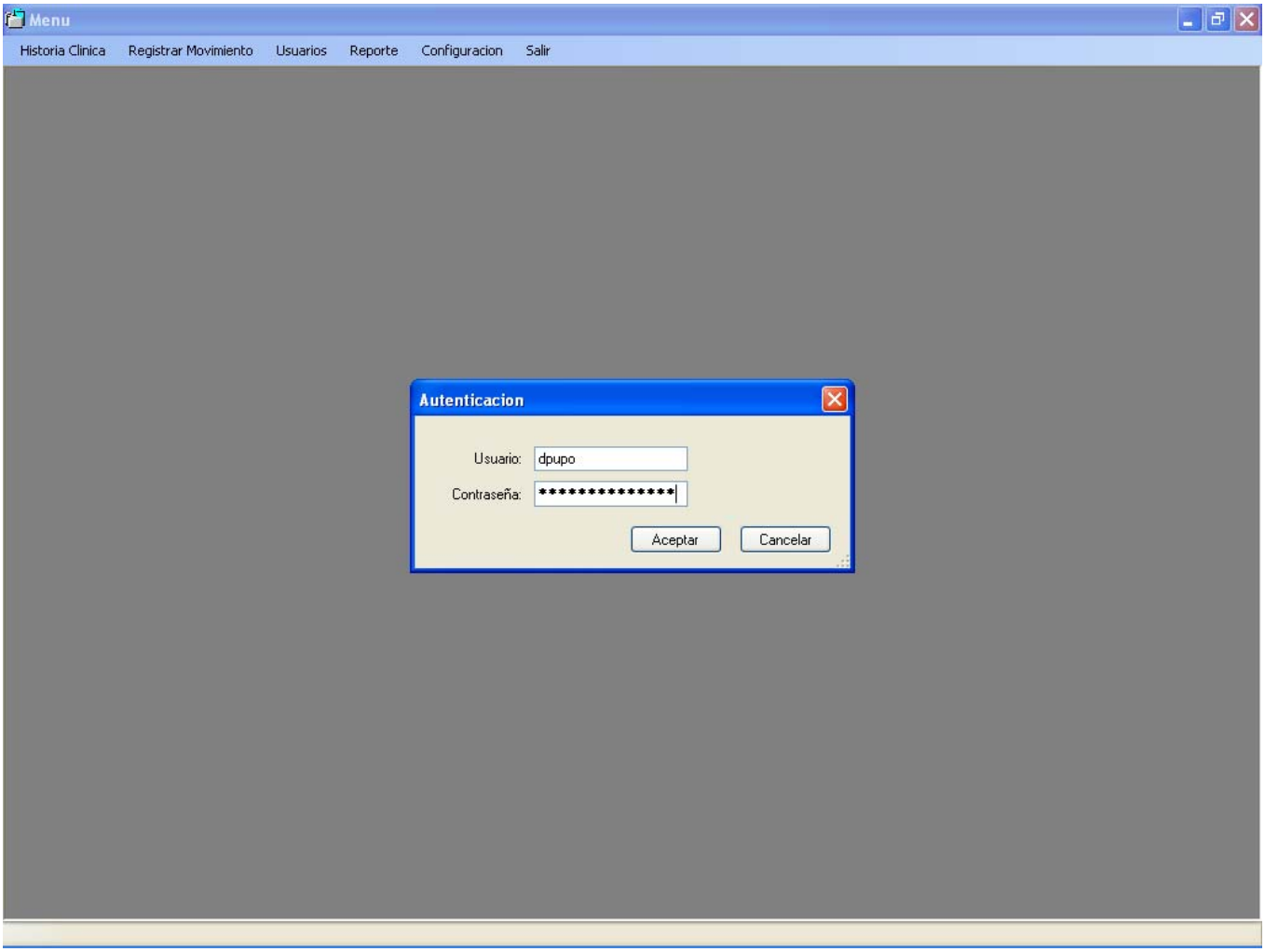
LOCKHART, T. *PostgreSQL Programmer's Guide*. Disponible en: http://www.screenart.es/documentacion/postgres/programmer/

NAVARRA, U. D. *Manual de introducción SQL -Centro de Tecnología Informática*, 1996. Disponible en: http://www.unav.es/cti/manuales/Intro_SQL/indice.html

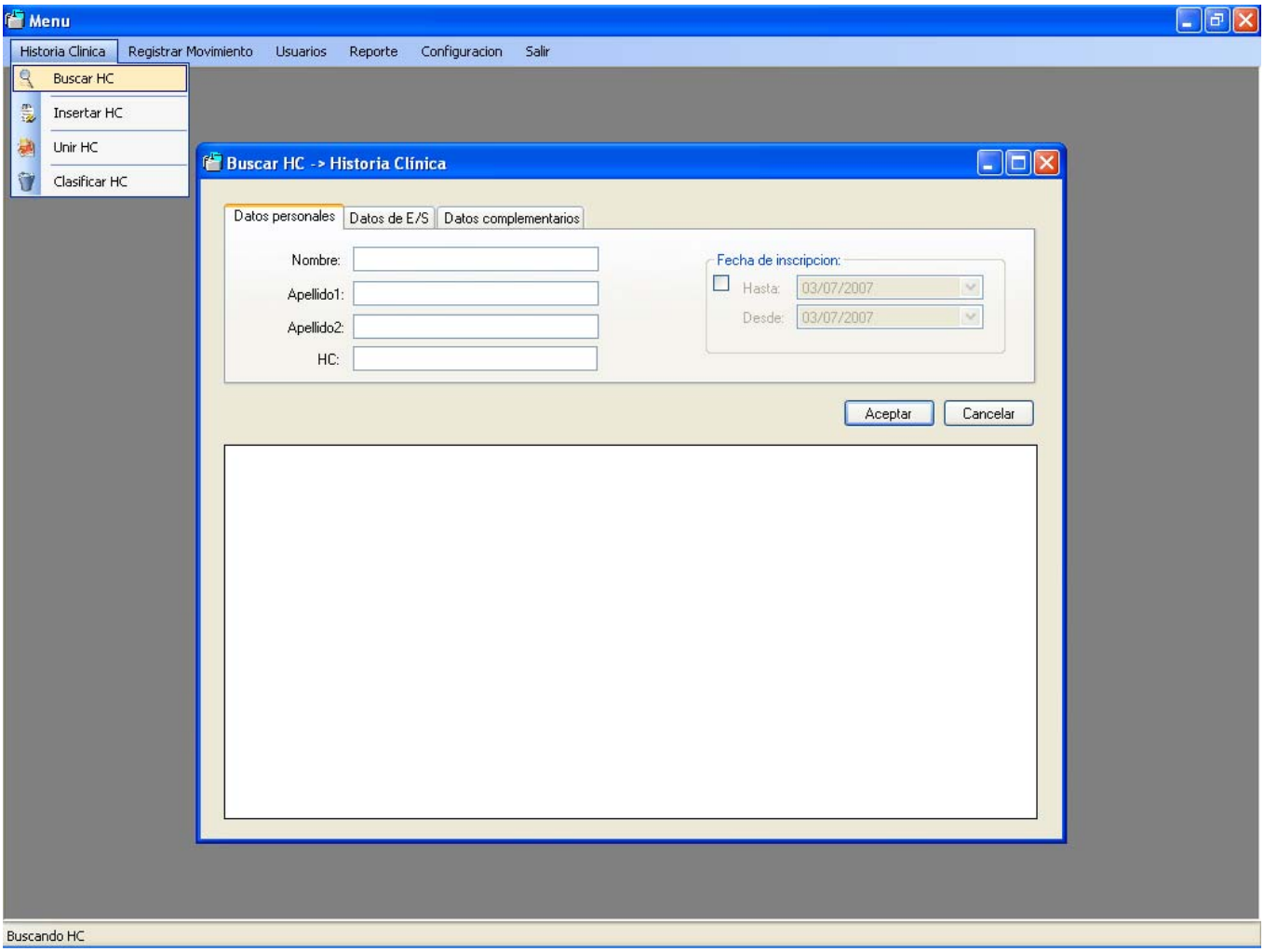
WIKIPEDIA, 2006]. Disponible en: http://es.wikipedia.org/

Anexo 1. Interfaz de la aplicación.

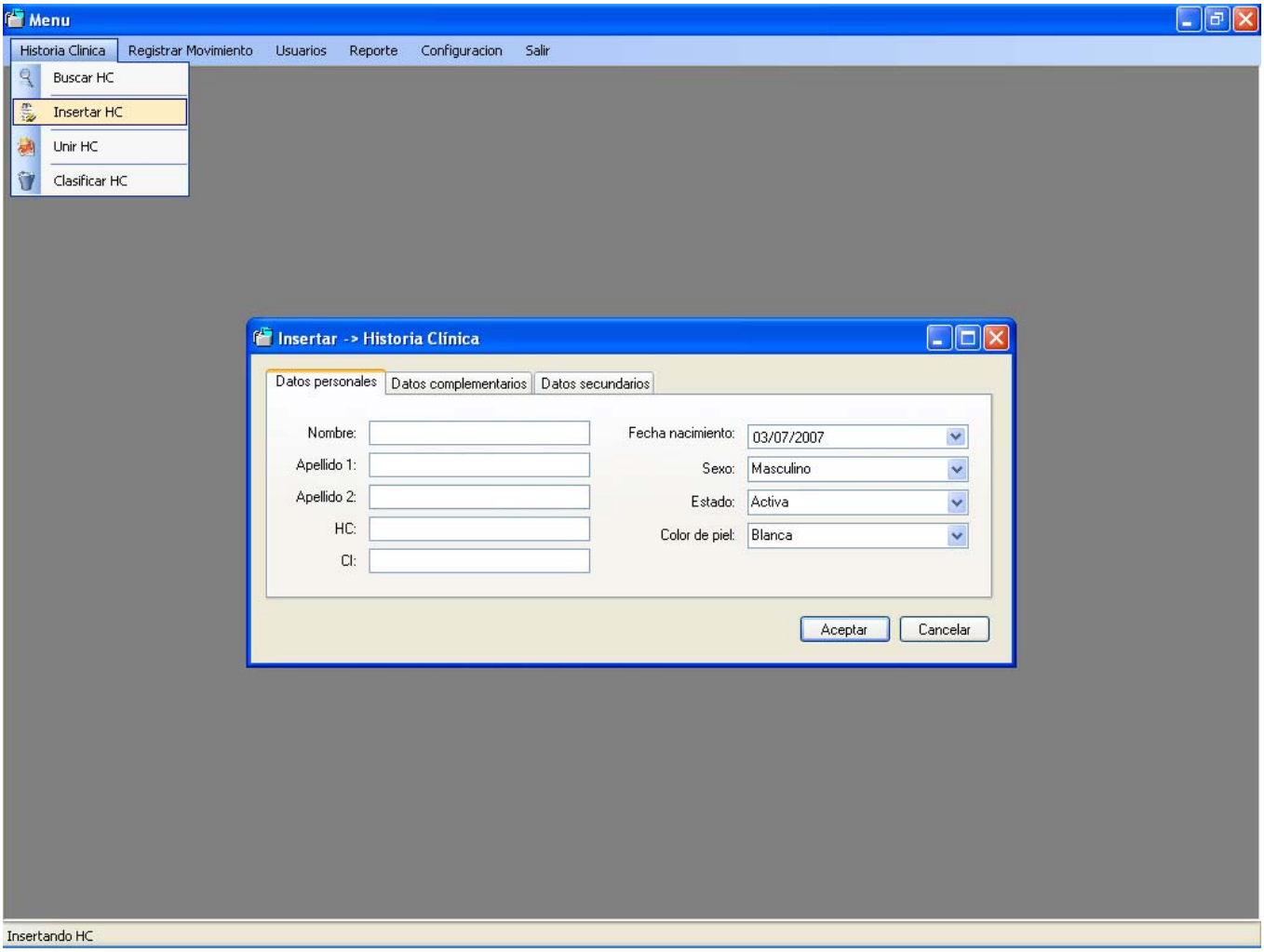
- **Autenticación.**



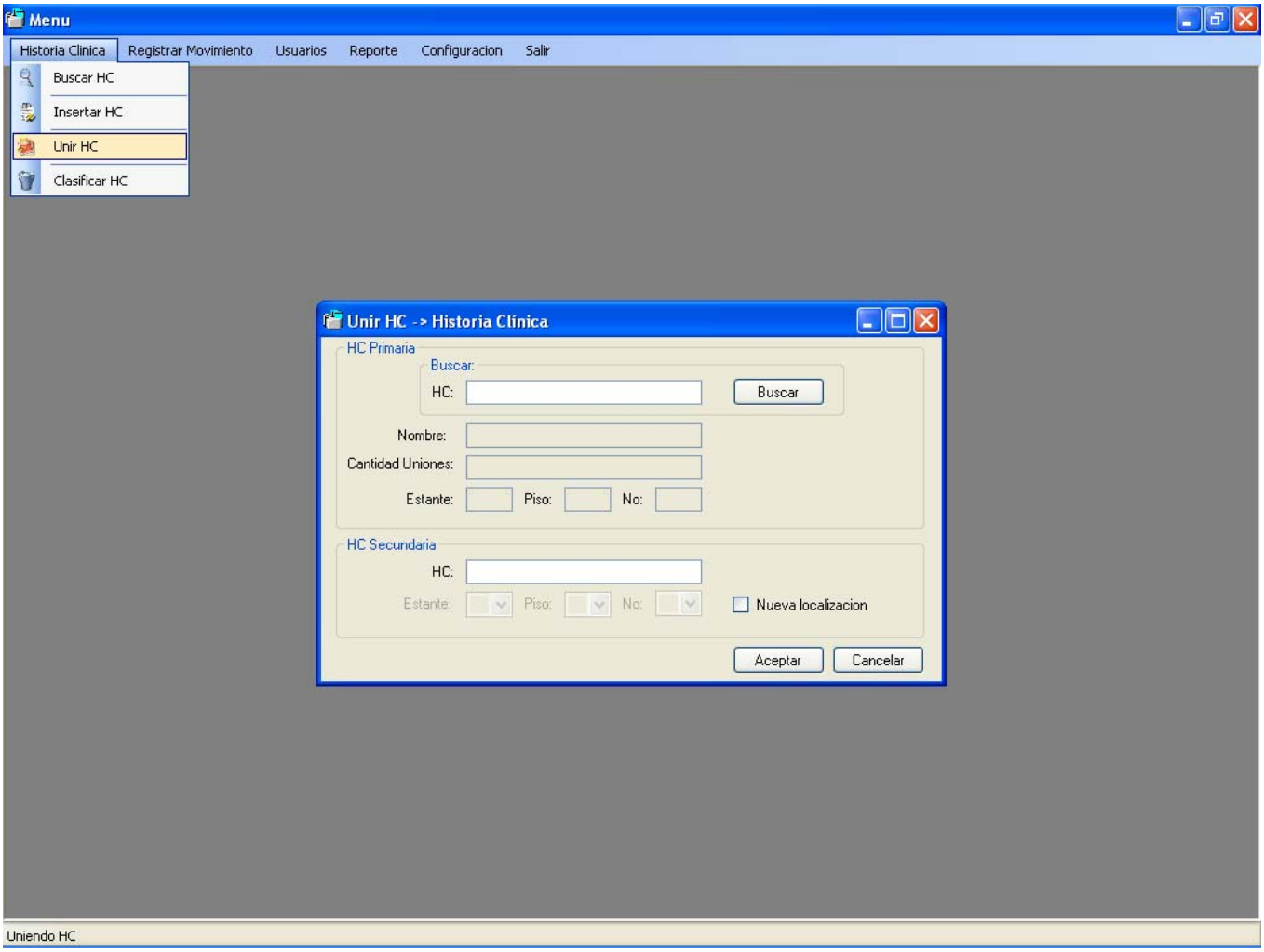
- **Buscar HC.**


Menu
Historia Clinica
Registrar Movimiento
Usuarios
Reporte
Configuracion
Salir
Buscar HC
Insertar HC
Unir HC
Clasificar HC
Buscar HC -> Historia Clínica
Datos personales
Datos de E/S
Datos complementarios
Nombre:
Apellido1:
Apellido2:
HC:
Fecha de inscripcion:
Hasta:
03/07/2007
Desde:
03/07/2007
Aceptar
Cancelar
Buscando HC

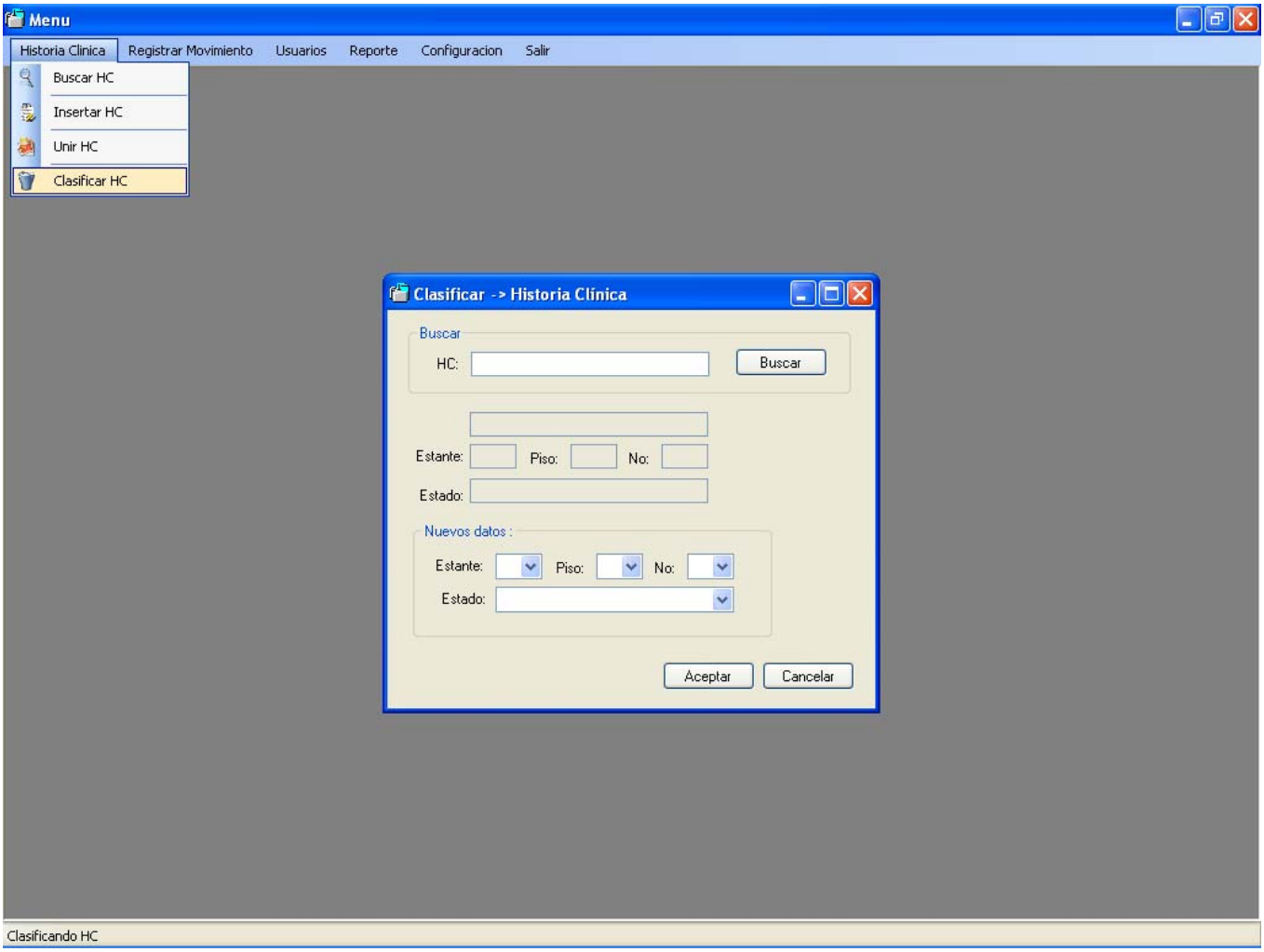
- **Insertar HC.**


Menu
Historia Clinica
Registrar Movimiento
Usuarios
Reporte
Configuracion
Salir
Buscar HC
Insertar HC
Unir HC
Clasificar HC
Insertar -> Historia Clínica
Datos personales
Datos complementarios
Datos secundarios
Nombre:
Apellido 1:
Apellido 2:
HC:
CI:
Fecha nacimiento:
03/07/2007
Sexo:
Masculino
Estado:
Activa
Color de piel:
Blanca
Aceptar
Cancelar
Insertando HC

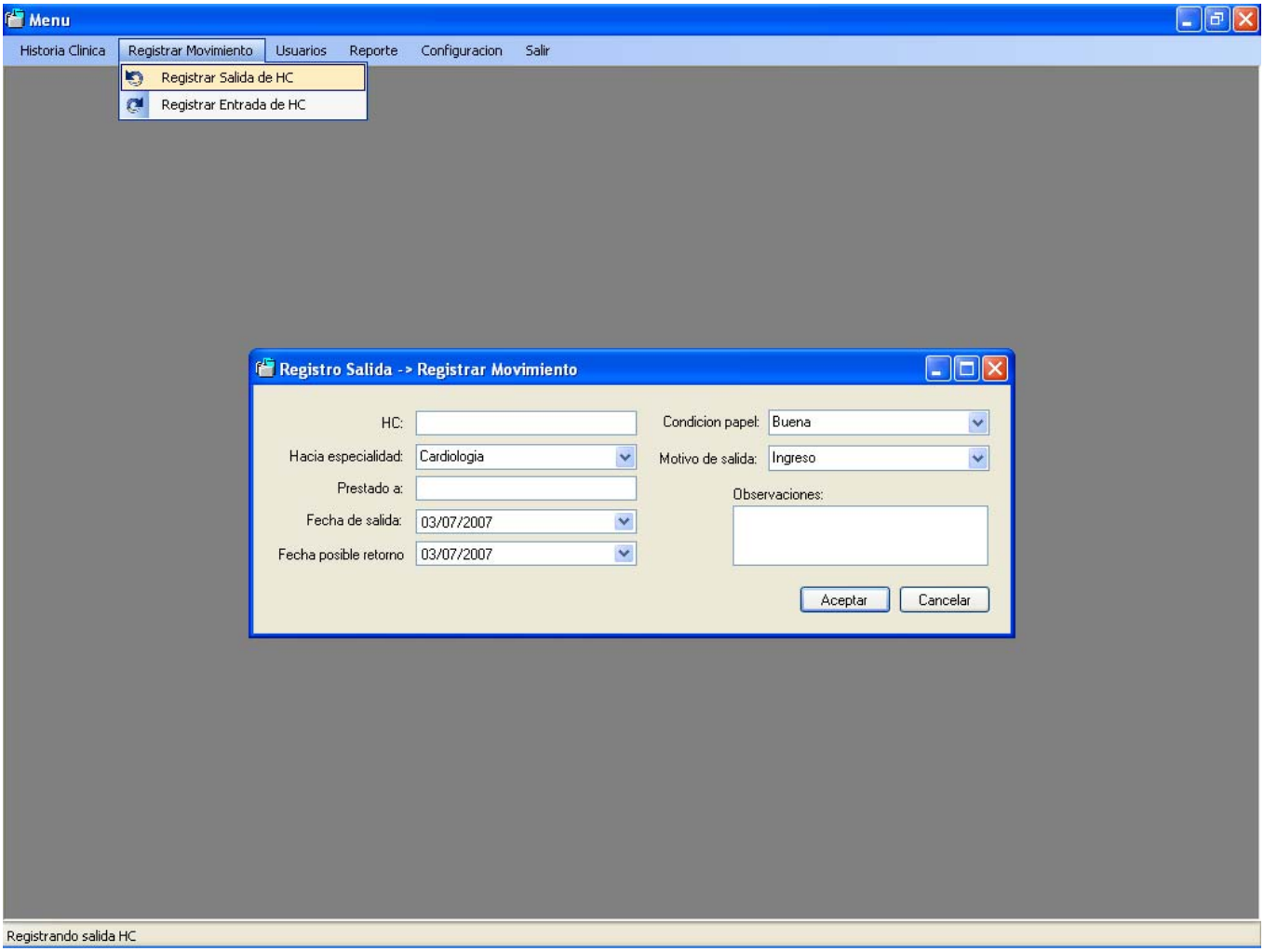
- **Unir HC.**


Menu
Historia Clinica
Registrar Movimiento
Usuarios
Reporte
Configuracion
Salir
Buscar HC
Insertar HC
Unir HC
Clasificar HC
Unir HC -> Historia Clínica
HC Primaria
Buscar:
HC:
Buscar
Nombre:
Cantidad Uniones:
Estante:
Piso:
No:
HC Secundaria
HC:
Estante:
Piso:
No:
Nueva localizacion
Aceptar
Cancelar
Uniendo HC

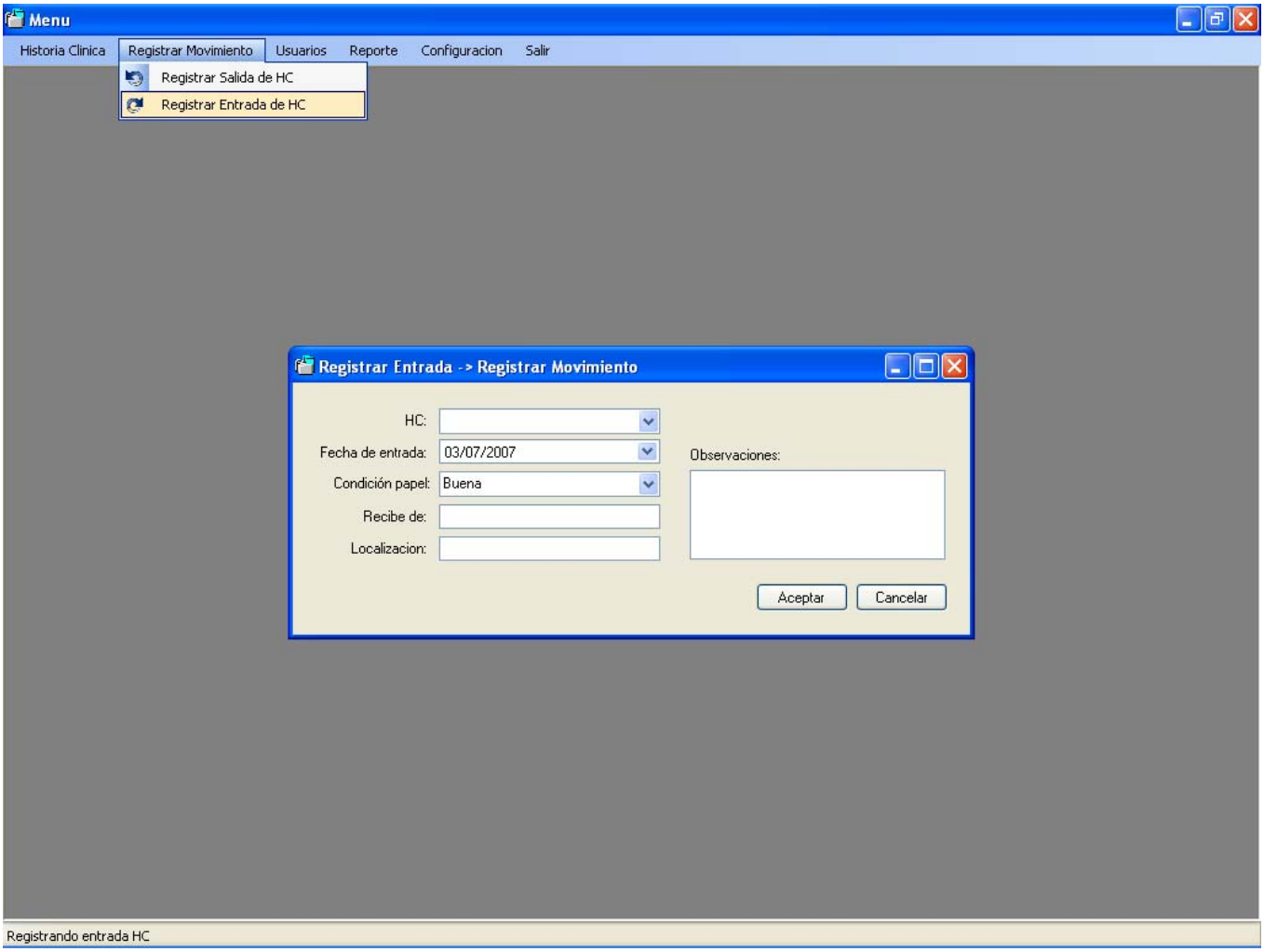
- **Clasificar HC.**



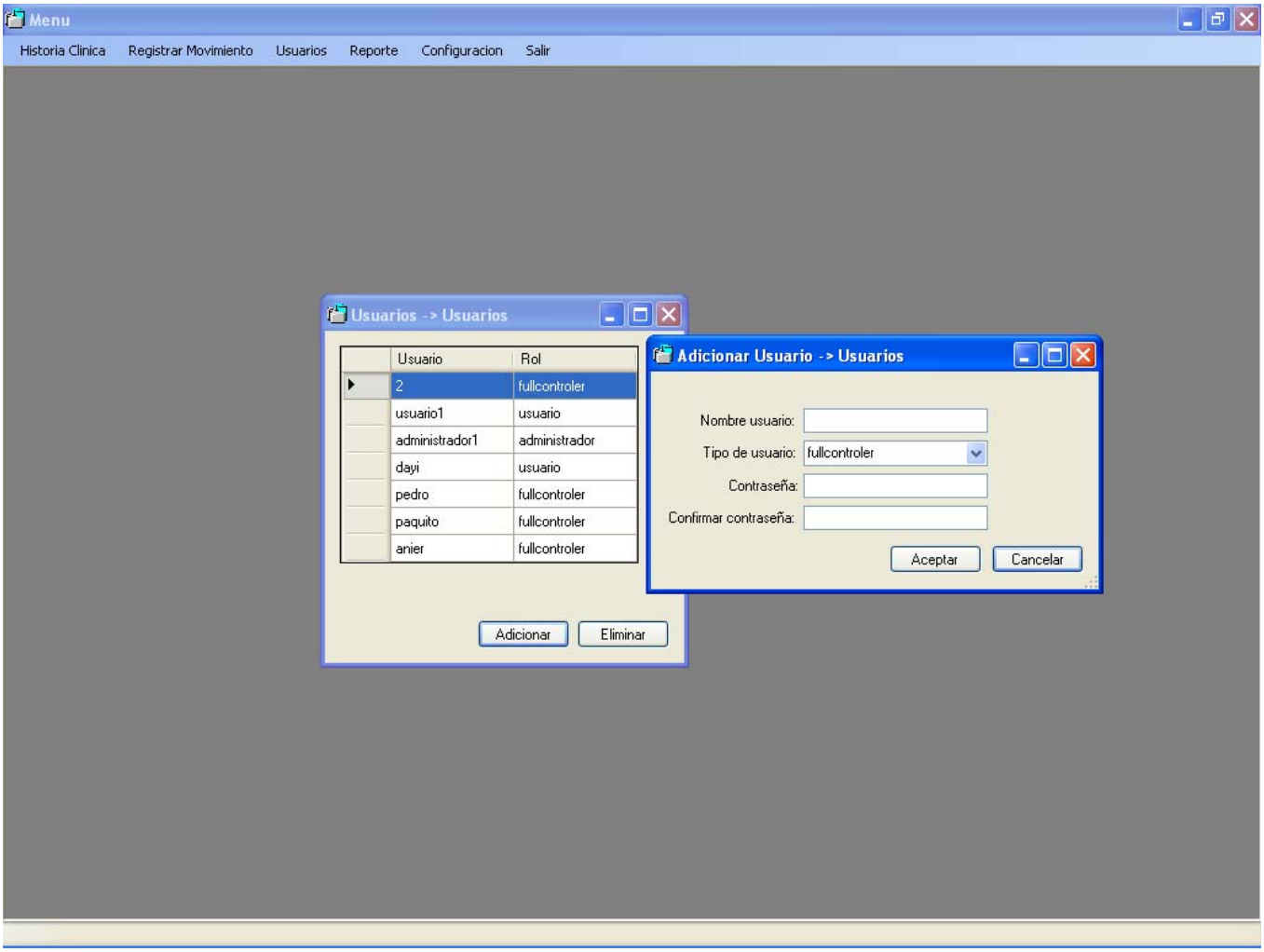
- **Registrar Salida de HC.**



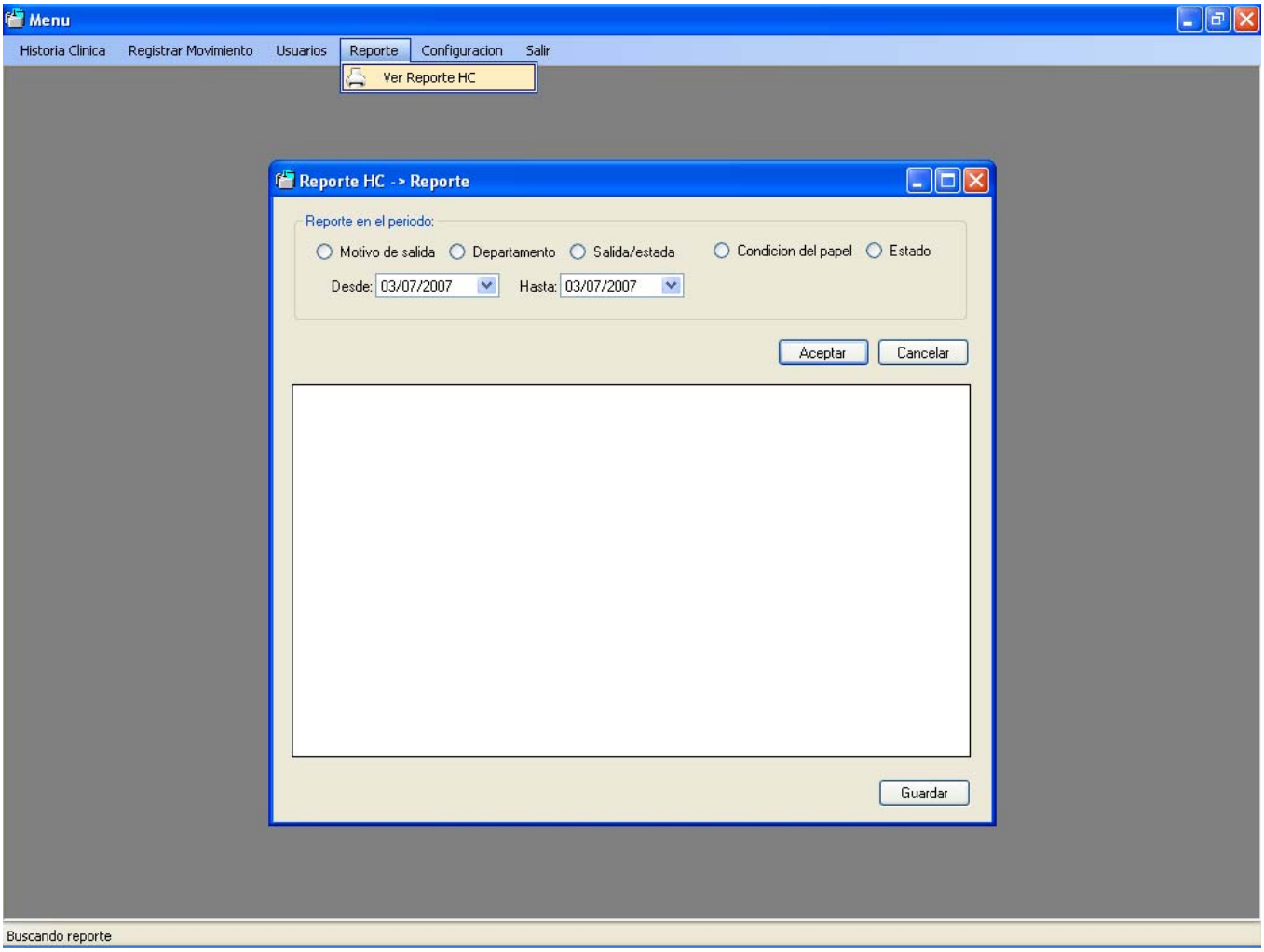
- **Registrar Entrada HC.**



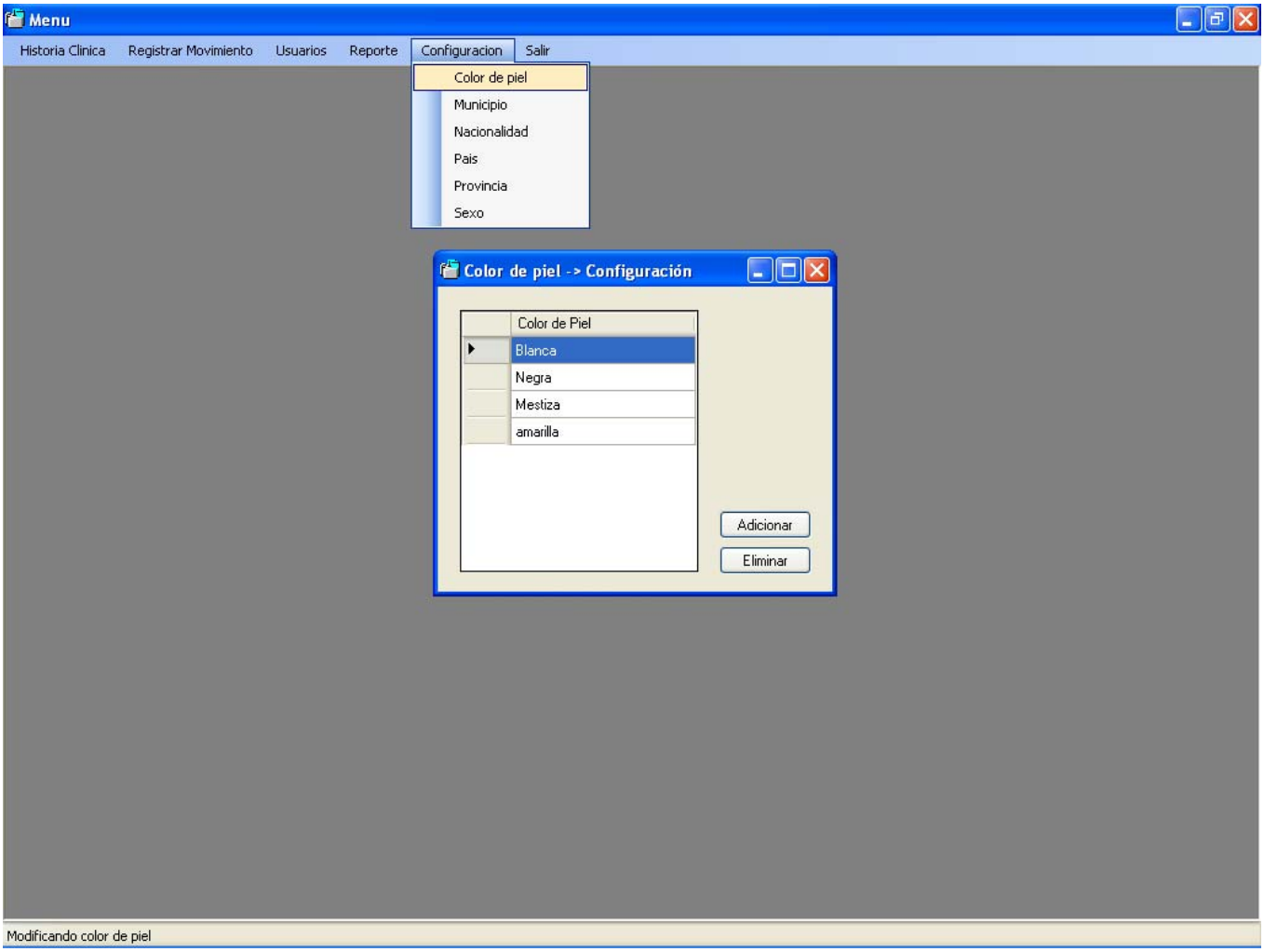
- **Adicionar y Eliminar Usuario.**



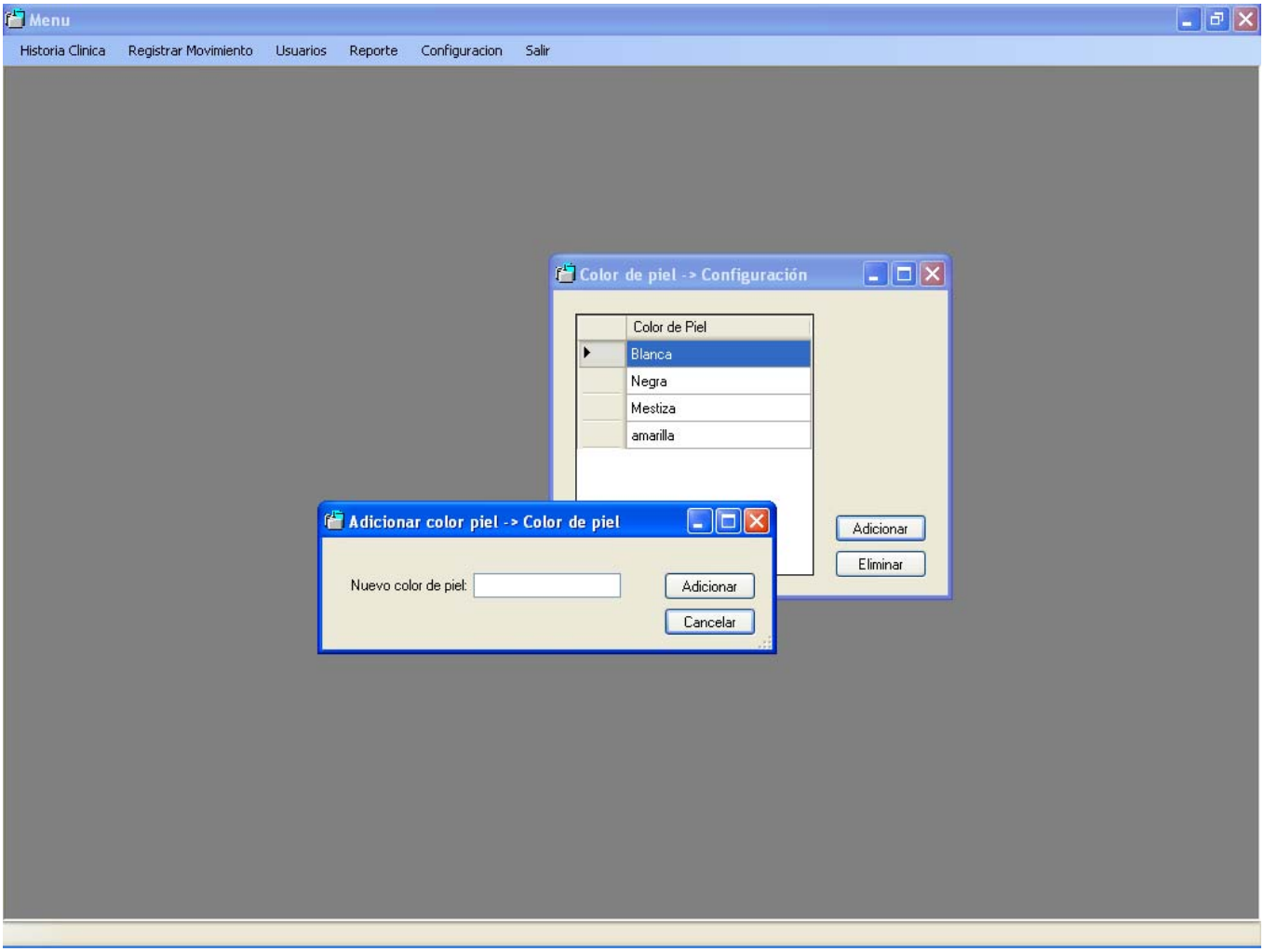
- **Ver Reportes de HC.**



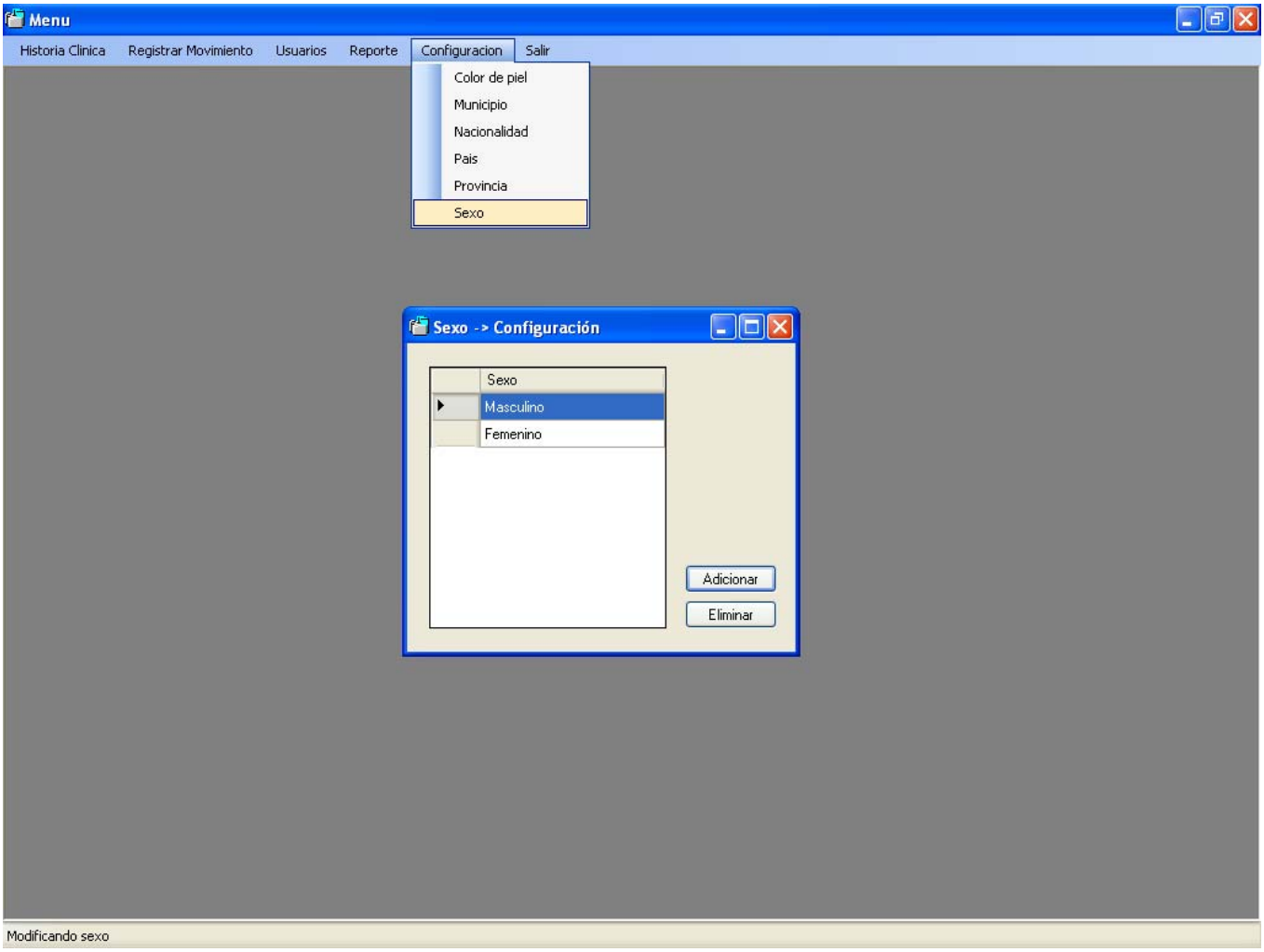
- **Configurar Color de Piel.**



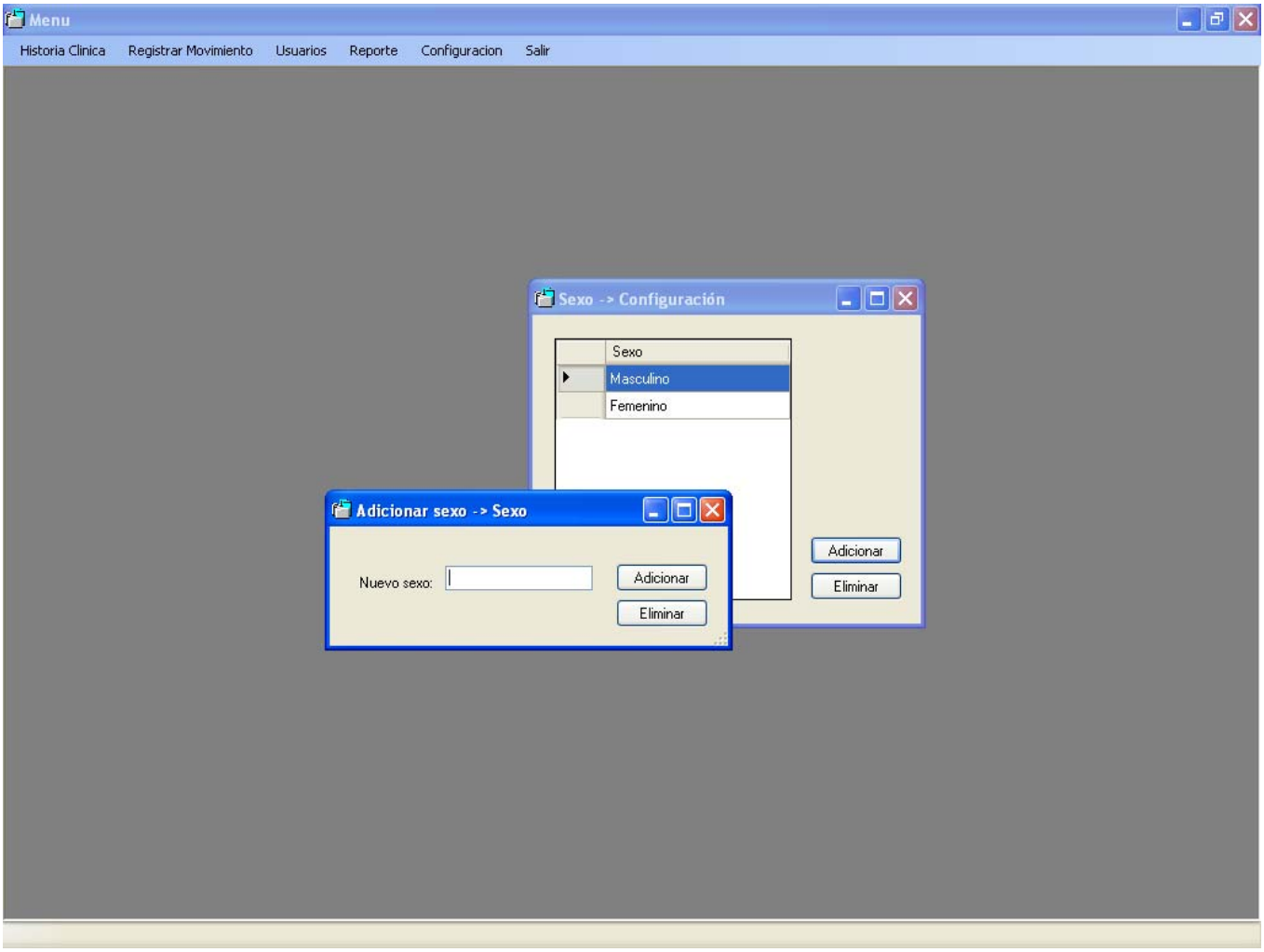
- **Adicionar nuevo Color de Piel.**



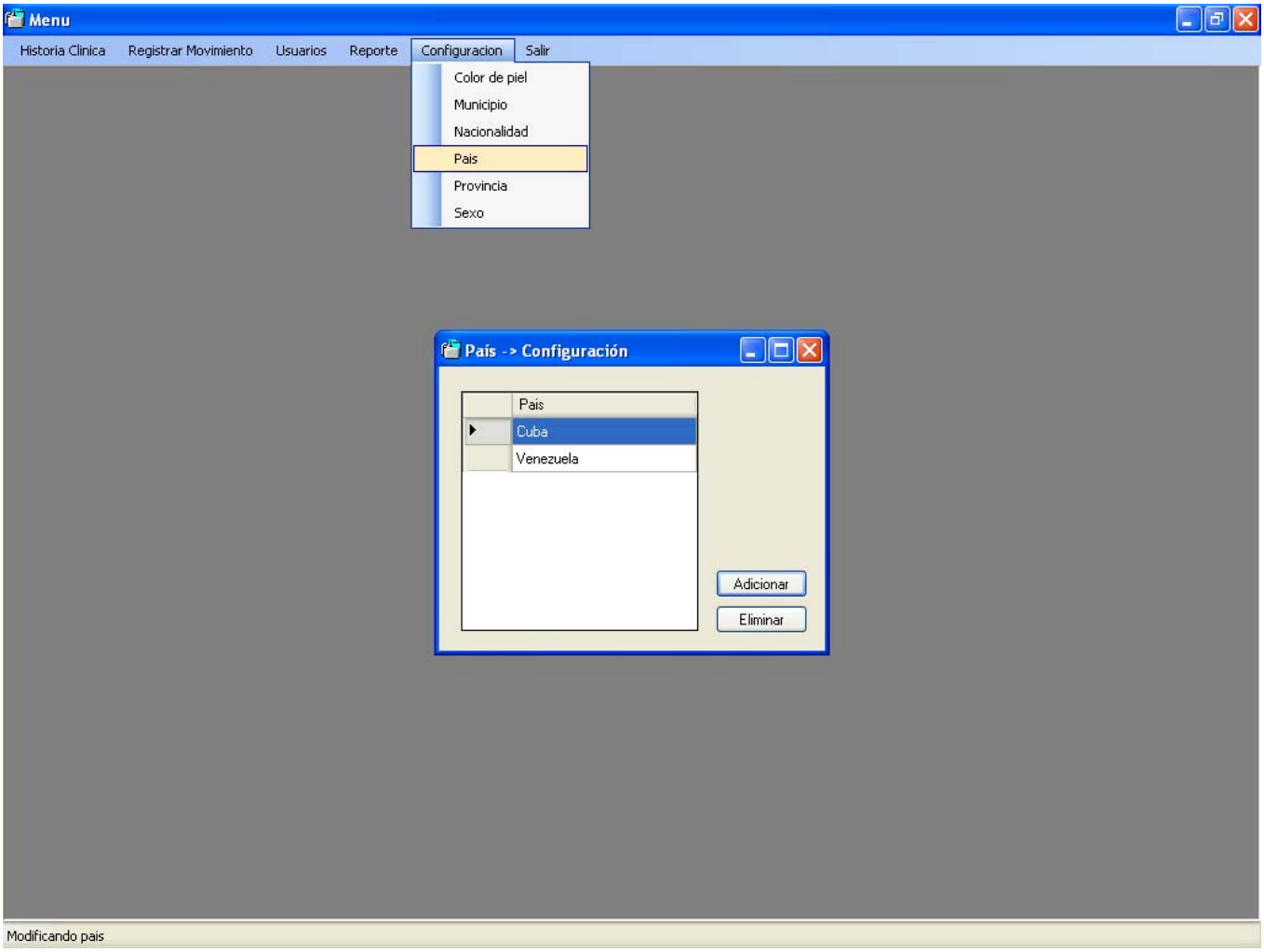
- **Configurar Sexo**



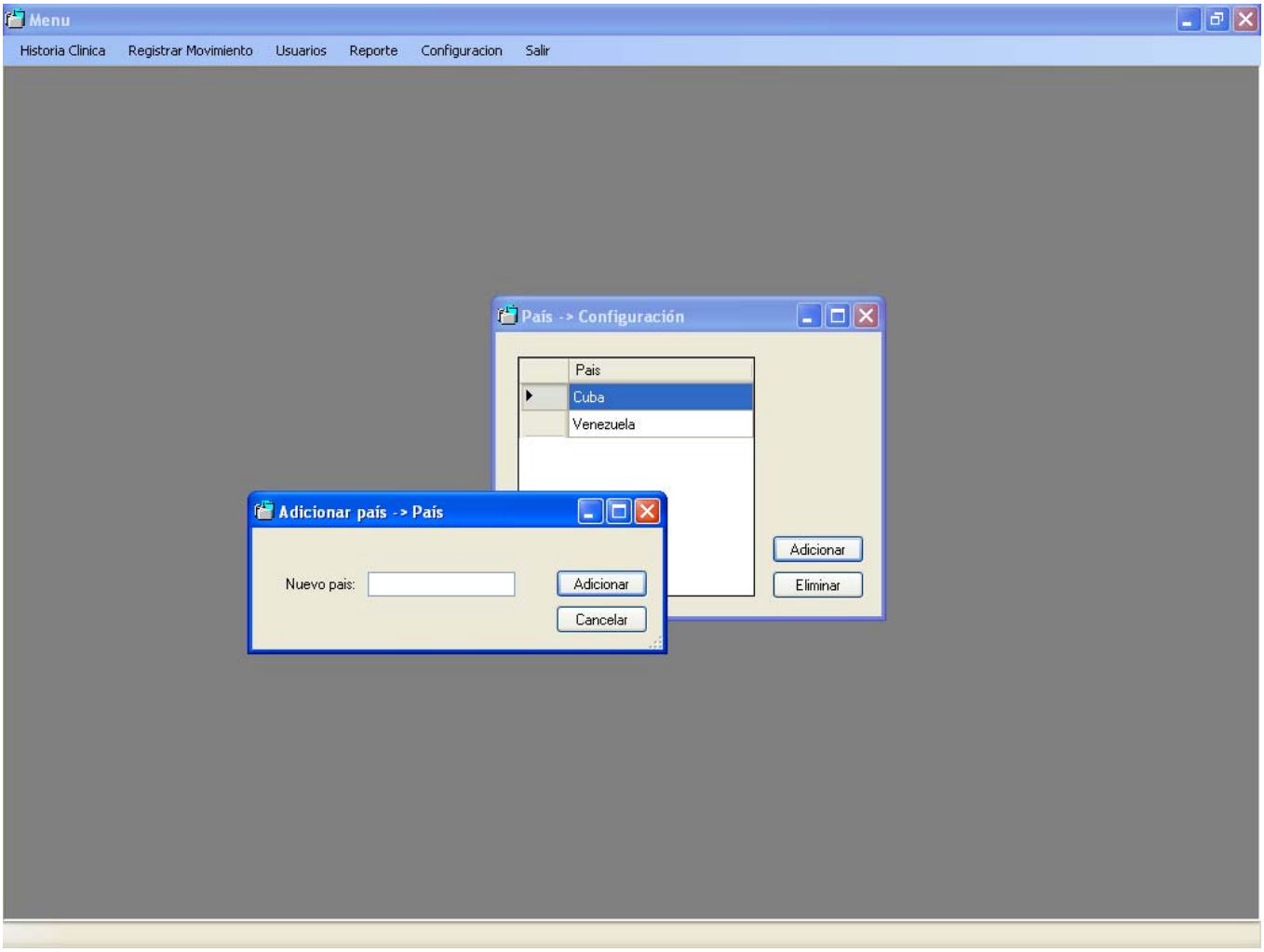
- **Adicionar nuevo Sexo.**



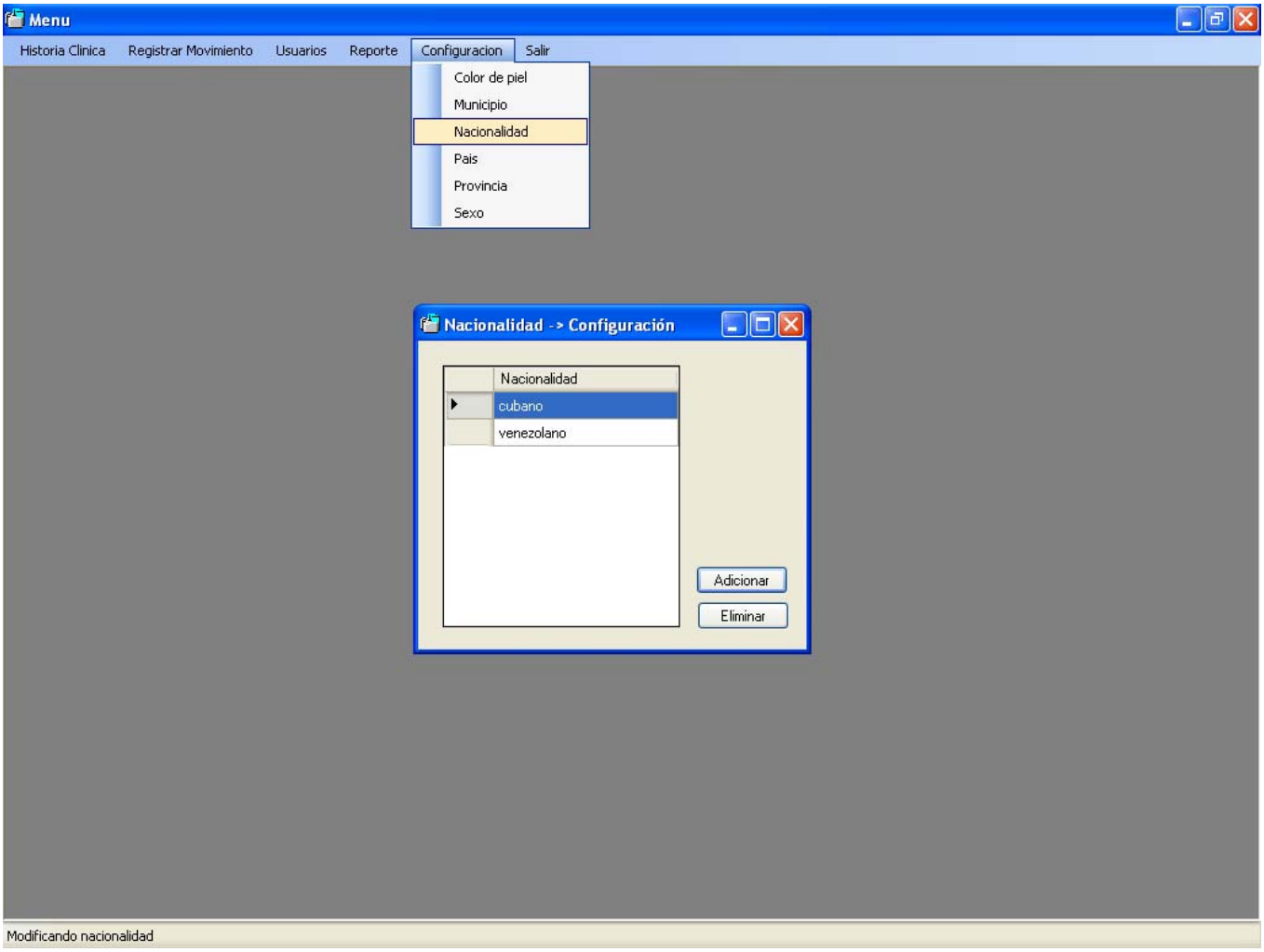
- **Configurar País.**



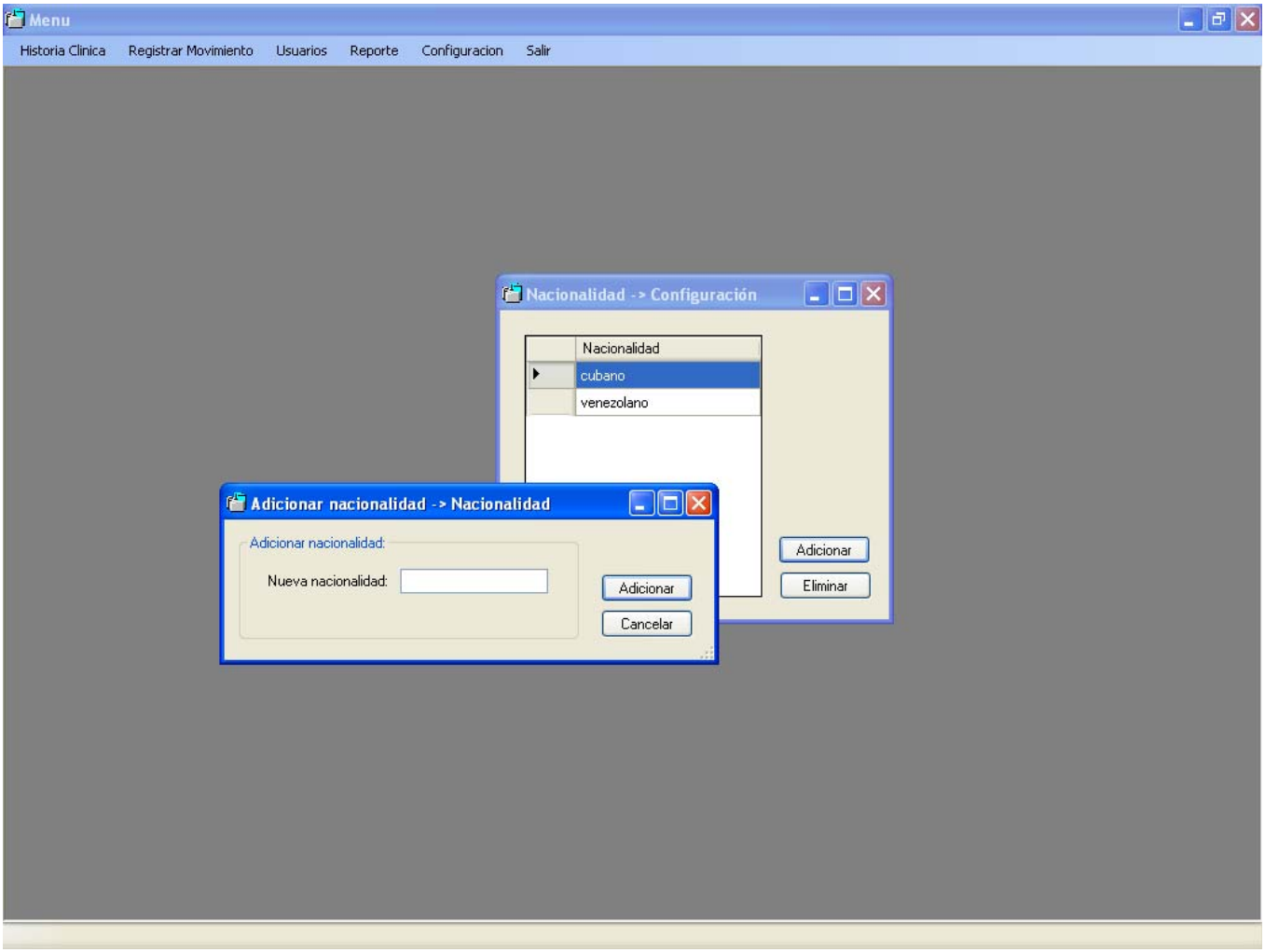
- **Adicionar nuevo País.**



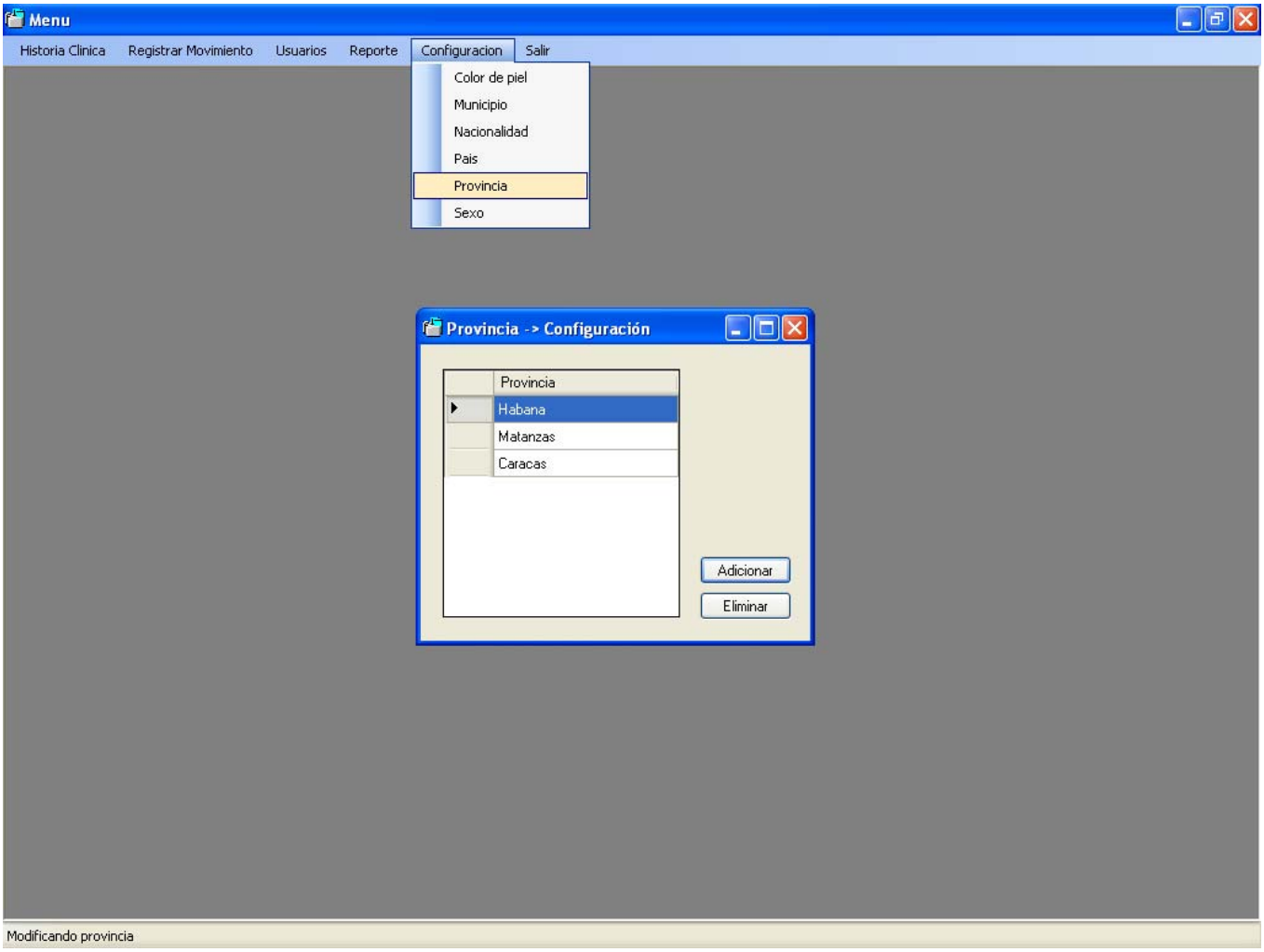
- **Configurar Nacionalidad.**



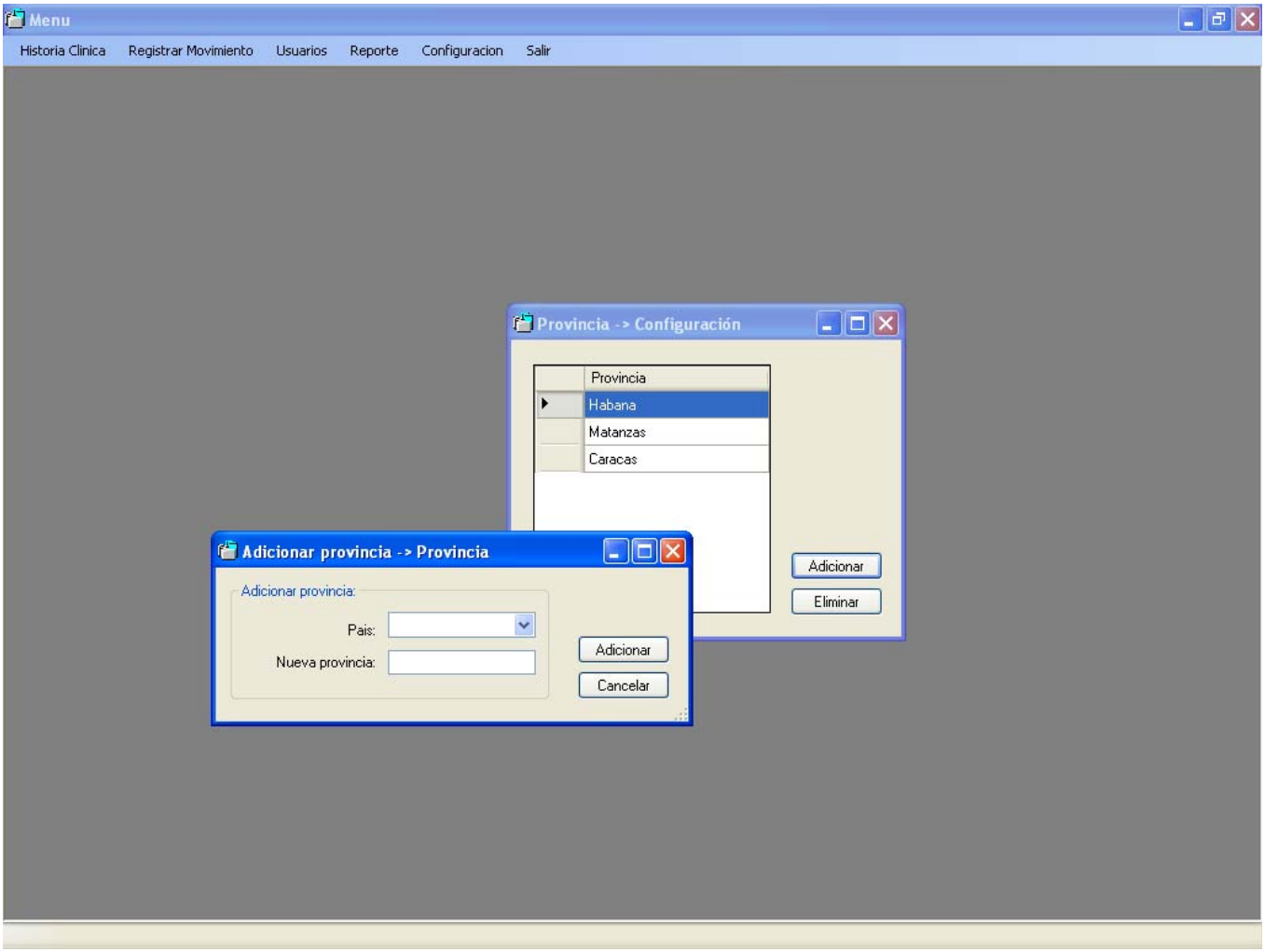
- **Adicionar nueva Nacionalidad.**



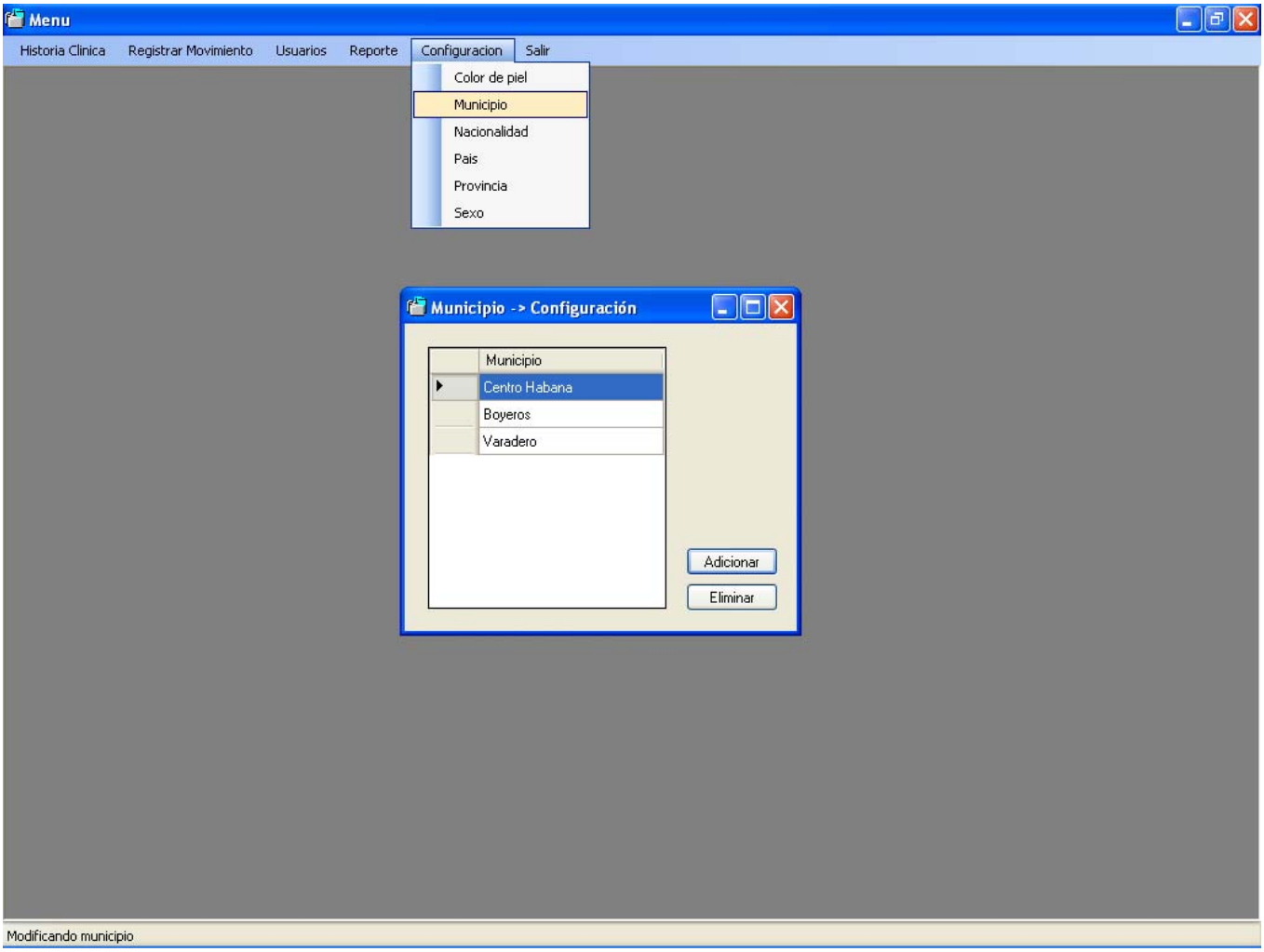
- **Configurar Provincia.**



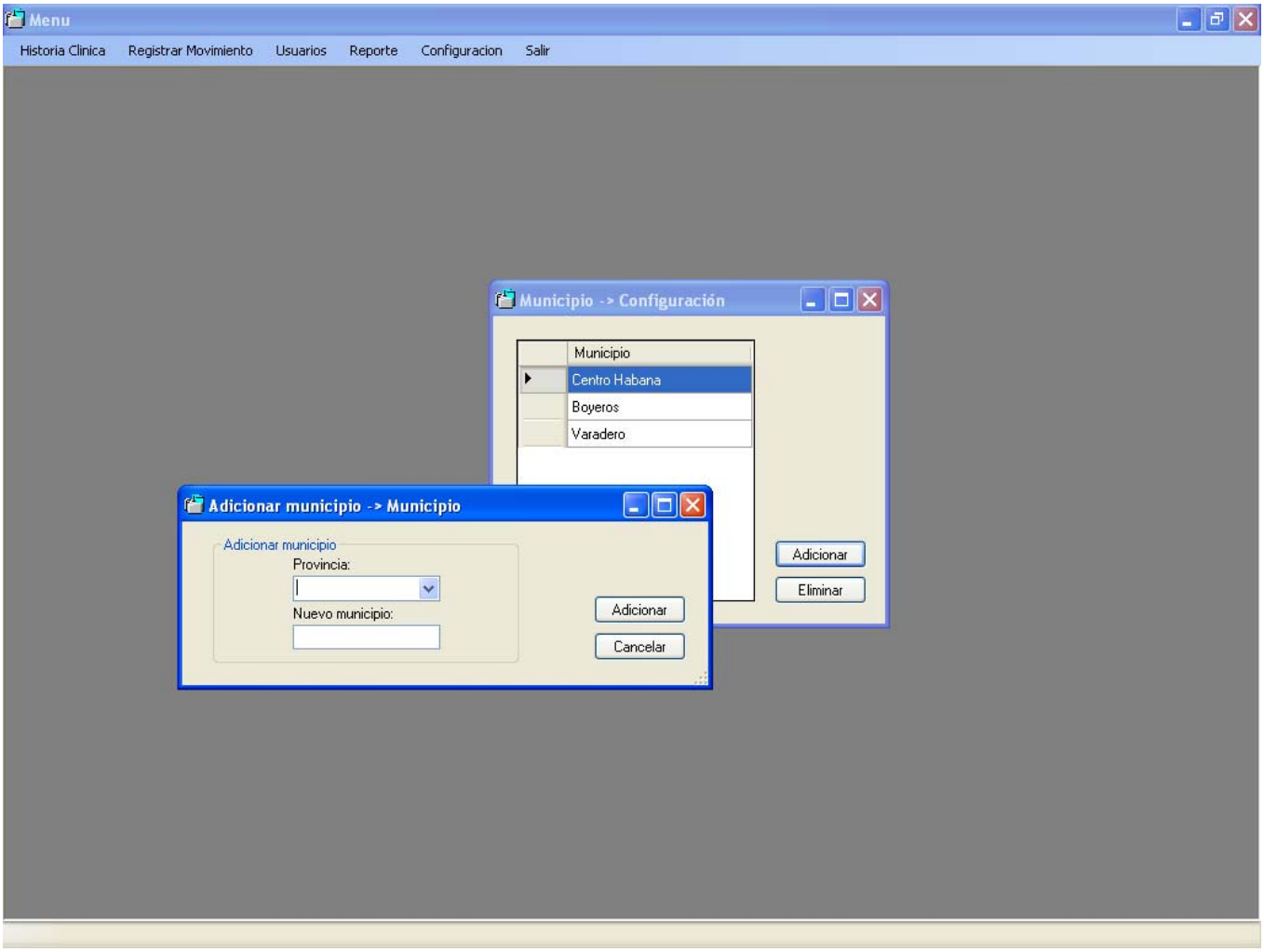
- **Adicionar nueva provincia.**



- **Configurar Municipio.**



- **Adicionar nuevo Municipio.**



Glosario de términos[2]

Base de datos: (database). Conjunto de datos relacionados que se almacenan de forma que se pueda acceder a ellos de manera sencilla, con la posibilidad de relacionarlos, ordenarlos en base a diferentes criterios, etc. [11]

Clases: Abstracciones que representan a un conjunto de objetos con un comportamiento e interfaz común. [12]

Clases abstractas: Una clase que declara la existencia de métodos pero no la implementación de dichos métodos. [13]

CLI: La infraestructura de lenguaje común (Common Language Infraestructure, CLI) es una especificación estandarizada que describe un entorno virtual para la ejecución de aplicaciones, cuya principal característica es la de permitir que aplicaciones escritas en distintos lenguajes de alto nivel puedan luego ejecutarse en múltiples plataformas tanto de hardware como de software sin necesidad de reescribir o recompilar su código fuente.

Compiladores: Un compilador es una aplicación que toma un programa escrito en un lenguaje de alto nivel y lo convierte a un programa escrito en un lenguaje que la maquina puede entender. [14]

Data warehouse: Un data warehouse es una colección de datos en la cual se encuentra integrada la información de una Institución y que se usa como soporte para el proceso de toma de decisiones gerenciales. [22]

ECMA: Organismo de estándares fundado en 1961. [23]

Frames: En castellano (marcos), son una manera de partir la página en distintos espacios independientes los unos de los otros, de modo que en cada espacio se coloca una página distinta que se codifica en un fichero HTML distinto. [15]

[2] Para elaborar este glosario fue utilizada la Wikipedia http://es.wikipedia.org y otras fuentes incluidas en las referencias bibliográficas.

GLOSARIO DE TERMINOS

Framework: En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Funciones: Conjunto de instrucciones que permiten procesar las variables para obtener un resultado. [16]

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. [17]

GNOME: es un entorno de escritorio para sistemas operativos de tipo Unix bajo tecnología X Window. Se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.

GNU: El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre: el sistema GNU.

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

GTK+: Es un grupo importante de bibliotecas o rutinas para desarrollar interfaces gráficas de usuario.

IDE: Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDE pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes.

Lenguaje interpretado: Un lenguaje de programación que fue diseñado para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados. También se les conoce como lenguajes de script.

Librerías: Una librería o biblioteca es un conjunto de procedimientos y funciones (subprogramas) agrupadas en un archivo con el fin de ser aprovechadas por otros programas.

Licencia LGPL: Es una licencia de software libre (LESSER GENERAL PUBLIC LICENCE). [18]

LGPL: (Lesser General Public License o Library General Public License), permiten el enlace dinámico de aplicaciones libres a aplicaciones no libres.

Orientado a objetos: Significa que el software se organiza como una colección de objetos discretos que contiene tanto estructura de datos como también un comportamiento. [19]

Plataforma de desarrollo: Es el entorno común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo, sin embargo, también es posible encontrarlas ligadas a una familia de lenguajes de programación o a una Interfaz de programación de aplicaciones o API por sus siglas en inglés.

Servidor: Es una computadora central, de gran capacidad, compartida por las otras computadoras de la red, llamadas Clientes o estaciones de trabajo (workstations), ya que reciben el servicio de almacenar, controlar y compartir la información contenida en el servidor. [20]

Sistemas operativos: Conjunto de programas que se integran con el hardware para facilitar al usuario, el aprovechamiento de los recursos disponibles. [21]