"Responsabilidad con pensamiento positivo"

UNIVERSIDAD TECNOLÓGICA ISRAEL

TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE:

INGENIERO/A EN SISTEMAS INFORMÁTICOS

TEMA: SOFTWARE PARA LA GESTIÓN DEL HISTORIAL CLÍNICO Y FACTURACIÓN.

AUTOR/ A: EDWIN SANTIAGO LÓPEZ LOJANO

ALEXANDER SANTIAGO PASTRANA TIPÁN

TUTOR/ A: Mgs HENRY RECALDE

QUITO- ECUADOR

AÑO: 2018

DECLARACIÓN Y

AUTORIZACIÓN

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación certifico:

Que el trabajo de titulación **"SOFTWARE PARA LA GESTIÓN DEL HISTORIAL CLÍNICO Y FACTURACIÓN"**, presentado por **Alexander Santiago Pastrana Tipán y Edwin Santiago López Lojano** estudiantes de la Carrera Ingeniería en Sistemas Informáticos, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito D. M., Septiembre del 2018

TUTOR

Ing. Henry Recalde, Mg

AGRADECIMIENTOS

Agradezco a la Universidad Tecnológica Israel la cual me abrió sus puertas permitiéndome prepararme para un futuro competitivo, a los docentes académicos de esta institución, quienes delicadamente contribuyeron con su tiempo, aportaron sus conocimientos y ofrecieron su apoyado durante cada etapa de mi formación académica, y de manera especial al magister Henry Recalde, tutor de nuestro proyecto de titulación, por su valiosa ayuda en asesorías y dudas presentadas en el desarrollo de la tesis.

DEDICATORIA

Dedico este proyecto a todas aquellas personas que, de alguna forma, son parte de del logro alcanzado, especialmente a mi familia, quienes a lo largo de mi vida han velado por mi bienestar y educación, depositando su entera confianza en mis capacidades y convirtiéndose en el motor impulso para avanzar con mis metas, y el pilar fundamental de apoyado y motivación para mi formación académica.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE GRÁFICOS

RESUMEN

El presente proyecto delinea el proceso de análisis, desarrollo e implementación de un sistema de gestión de historial clínico y facturación, diseñado en Visual Studio 2010, SQL Server 2008 y UI DevExpress, con el cual se proyecta optimizar la administración del consultorio médico Salud & Vida, por medio de procesos sistematizados agiles y precisos que permitan mantener un adecuado control y servicio médico al paciente, facilitando el trabajo y reduciendo el tiempo de atención.

Por medio de la implementación del sistema se logró regular y simplificar los procesos de agendamiento de citas y de las historias clínicas de los pacientes, mantener un registro adecuado de la información y minimizar los tiempos de atención a los pacientes.

Para la ejecución del proyecto se adoptó la metodología de desarrollo Ágil XP conocida por sus siglas XP (**eXtreme Programming),** una metodología de rápida implementación que permite el control de iteraciones en periodos cortos de tiempo.

En base a los resultados obtenidos producto de la implementación del proyecto, se validó la reducción del tiempo empleado en menos 20 minutos y la mejora en el manejo y administración de la información recabada de pacientes, facilitando la gestión del médico y mejora en el servicio que brinda el consultorio médico.

PALABRAS CLAVES

Visual Studio 2010, SQL Server 2008, DevExpress, XP, UI

ABSTRACT

This project delineates the process of analysis, development and implementation of a management system of clinical history and billing, designed in Visual Studio 2010, SQL Server 2008 and UI DevExpress, with which it is planned to optimize the management of the medical office Salud & Vida, by means of agile and precise systematized processes that allow to maintain an adequate control and medical service to the patient, facilitating the work and reducing the time of attention.

Through the implementation of the system, it was possible to regulate and simplify the appointment scheduling processes and the patients' medical records, maintain an adequate record of the information and minimize patient care times.

For the execution of the project, the Agile XP development methodology known by its acronym XP (eXtreme Programming) was adopted, a rapid implementation methodology that allows the control of iterations in short periods of time.

Based on the results obtained from the implementation of the project, the reduction in time spent in less than 20 minutes was validated and the improvement in the management and administration of the information collected from patients was validated, facilitating the management of the doctor and improving the service provided by the doctor's office.

PALABRAS CLAVES

Visual Studio 2010, SQL Server 2008, DevExpress, XP, UI

INTRODUCCIÓN

Antecedentes

El consultorio médico Salud & Vida se dedica a la prestación de servicios de salud, actualmente el proceso es ineficiente debido a que la información del paciente que se deriva de la consulta médica, así como el proceso de facturación que conlleva se lo realiza de forma manual (documento físico), dificultando tanto la visualización del historial clínico de cada paciente para las atenciones subsecuentes, como la emisión de la factura correspondiente, por tanto genera un incremento de tiempo no estimado por atención, ocasionando incomodidad en los pacientes por el tiempo de espera.

Planteamiento del problema

El proceso de recabar la información del paciente es vulnerable por no ser esta almacenada en una base de datos, problema que conlleva no poseer un adecuado control estadístico de los pacientes que son atendidos por diferentes patologías, y que a su vez no permita brindar una atención preventiva.

Al término de la atención del paciente se emite una factura realizada manualmente, lo cual ocasiona diversos errores al totalizar la suma de valores. Por estos inconvenientes surge la necesidad de desarrollar un software que permita dar una solución al problema y optimizar las necesidades que demanda la empresa de salud.

Formulación del Problema

Actualmente el consultorio clínico realiza sus procesos de historial clínico, agendamiento de citas y facturación de forma manual, por tal motivo el volumen de información se convertido en un problema al momento del manejo y manipulación de la misma, afectando la calidad de la atención.

El presente trabajo tiene por objetivo desarrollar un sistema web de gestión de historial clínico y facturación, mediante el cual se podrá ingresar y consultar los datos del paciente,

realizar el agendamiento de citas médicas, llevar el control e historial clínico del paciente, así como la emisión de facturas.

El desarrollo del sistema pretende lograr una mejora en el registro médico del paciente y disminuir el tiempo de espera de los pacientes, por tanto se espera facilitar el proceso con la creación de una aplicación web que ayude a la gestión y administración del consultorio clínico, por medio de procesos sistematizados agiles, precisos y sin errores como los generados en los procesos manuales.

Justificación

La importancia de desarrollar un sistema de gestión de historial clínico y facturación, es de gran valor para la administración y cometido del consultorio médico Salud & Vida, debido a que permite mantener un adecuado control y servicio médico al paciente, facilitando el trabajo y optimizando el tiempo de atención.

Objetivo General.

Desarrollar e implementar un sistema para la gestión y administración de citas médicas, historial clínico de los pacientes y facturación para el consultorio médico Salud y Vida, mediante la creación de un software diseñado en Visual Studio 2010, SQL Server 2008 y UI DevExpress, que permita el adecuado registro, almacenamiento y administración de la información derivada de las consultas médicas.

Objetivos Específicos.

- Analizar y determinar las necesidades del usuario.
- Estudiar y redefinir los procesos actuales del consultorio médico.
- Desarrollar un sistema de información en lenguaje de programación C#, diseñar interfaces gráficas y modelo de base de datos acorde a las necesidades del consultorio clínico.
- Realizar pruebas para determinar un correcto funcionamiento del sistema.

- Implementar y capacitar a los usuarios sobre el manejo del sistema.

Descripción de los Capítulos

Capítulo 1. Fundamentación Teórica.- Se describe de forma conceptual las herramientas que se utilizaran en el ciclo de vida de implementación del proyecto integrador.

Capítulo 2. Marco Metodológico.- Se describe el tipo de estudio que se utiliza en el proceso de investigación, la metodología seleccionada, la descripción de la aplicación de las herramientas para la consecución del proyecto.

Capítulo 3. Propuesta.- Se describe el detalle de la solución propuesta, la explicación del proceso actual por medio de un diagrama general del proceso, la explicación de la automatización por medio de un diagrama propuesto, requerimientos necesarios, el alcance que tendrá el proyecto, las restricciones, la descripción de pruebas necesarias para la validación.

Capítulo 4. Implementación.- Se describe de forma detallada la ejecución de la propuesta, se detalla el diseño de la interfaz, el modelo físico de la Base de Datos, la arquitectura utilizada, los estándares de programación a utilizarse. La pruebas que se ejecutan para validar el producto.

Capítulo 5-6. Conclusiones y Recomendaciones.- Se describen las novedades que se presentan en la puesta en marcha del proyecto con el fin de generar recomendaciones para un buen funcionamiento de la solución implementada.

1. CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.1 Programación.

Para el desarrollo de programas de cierta envergadura o complejos, con ciertas garantías de calidad, es conveniente seguir alguno de los modelos de desarrollo de software existentes, en donde la programación es sólo una de las etapas del proceso de desarrollo de software. Los modelos de desarrollo de software abordan una disciplina específica del campo de la informática. (Concepto de Programación, 2018)

Mediante la creación del sistema para la consulta externa y facturación de ventas del consultorio "Médico Vida & Salud" se logrará mejorar los procesos actuales obteniendo como resultado la disminución del tiempo de espera del paciente y mejorar el servicio de atención de los pacientes que acuden al consultorio.

1.2 Programación orientada a objetos

La programación a objetos promete mejoras amplias en la forma de diseño, desarrollo y mantenimiento del software ofreciendo soluciones a largo plazo a los problemas e inconvenientes que han existido desde el inicio en el desarrollo de software: como falta de portabilidad y la reusabilidad de código que son difíciles de modificar, técnicas de codificación no intuitivas y ciclos de desarrollo largos (Harvey, 2007)

Un lenguaje orientado a objetos combate estos problemas. Por tanto tiene tres características básicas: basado en objetos, basado en clases y sobre todo debe ser capaz de tener herencia de clases. Existen lenguajes que cumplen uno o dos de estos puntos; mucho menos existe lenguajes que cumplen las tres características básicas. La meta más difícil de sortear es usualmente la herencia. (Harvey, 2007)

1.2.1 Lenguaje de programación C#

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando estos últimos años con el objetivo de mejorar tanto su sistema operativo como su

modelo de componentes (COM) para obtener una plataforma con la que sea sencillo el desarrollo de software en forma de servicios web.

Los servicios web son un novedoso tipo de componentes software que se caracterizan a la hora de trabajar por su total independencia respecto a su ubicación física real, la plataforma sobre la que corre, el lenguaje de programación con el que hayan sido desarrollados o el modelo de componentes utilizado para ello.

1.2.2 AspNet

ASP.NET es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web empresariales con el código mínimo. ASP.NET forma parte de .NET Framework y al codificar las aplicaciones ASP.NET tiene acceso a las clases en .NET Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR), entre ellos Microsoft Visual Basic, C#, JScript .NET y J#. Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del Common Language Runtime, seguridad de tipos, herencia, etc. (MICROSOFT, 2007)

1.2.3 JQuery

JQuery es una librería JavaScript open source que funciona en múltiples navegadores, y que es compatible con CSS3. Como principal objetivo principal es generar la programación de los scripts mucho más ágil y sencilla por parte del cliente. Con jQuery se pueden producir en un tiempo relativamente corto páginas dinámicas así como también animaciones parecidas a flash, minimizando de esta manera el tiempo que es un factor muy importante en el desarrollo. También al ser jQuery actualmente una de las mejores librerías en el mundo JavaScript, permite agregar fácilmente plugins, ahorrando de manera significativa esfuerzo y tiempo que es uno de los principales motivos por los que se creó esta librería. Esta librería es rápida y flexible en el desarrollo web. (LANCKER, 2014)

1.2.4 C#

C#', el cual se denomina *si sharp* en inglés y *c sharp* en español, es un lenguaje de programación orientado a objetos, desarrollado por Microsoft y contenido dentro de .Net. Su sintaxis es similar a Java y actualmente se ha convertido en un lenguaje tremendamente flexible. (Duran, 2007)

- El soporte para múltiples paradigmas de programación.
- El ser multiplataforma Windows, Unix, Android, iOS.
- El permitir desarrollar todo tipo de aplicaciones:
    - Aplicaciones de escritorio, en consola o con interfaz gráfica con WPF.
    - Aplicaciones para dispositivos móviles con Xamarin.
    - Aplicaciones y páginas web con ASP.NET

1.2.5 Staments (declaraciones)

En un programas de lenguaje C# se definen a las declaraciones como todos los elementos que constituyen al programa desarrollado.

1.2.6 Clases y Objetos

Son la operación fundamental del lenguaje C#, ya que esta posee datos en diferentes estados (fields), y las acciones que puede tener (métodos), en una sola unidad u objeto. Además, provee la facilidad de crear instancias de las mismas que son conocidos como objetos, también permiten la herencia y polimorfismo, cuando una clase deriva una de la otra, también se pueden extender de clases base.

1.2.7 Members

Los miembros de la clase puedes ser de una estática (static) o de una instancia. Pertenecen a una clase los miembros estáticos y pertenecen a la instancia de la clase los miembros de instancia.

1.2.8 Internet Information Service (IIS)

El IIS son servicios para servidores de Microsoft, especialmente usado en servidores web, viene integrado con Windows NT 4.0. Es fácil de administrar por estar ligado al sistema operativo. (Definición de Servidor IIS, s.f.)

Los requisitos de hardware van de acuerdo a las exigencias específicas del servidor para un funcionamiento óptimo del IIS, también de la cantidad aproximada de usuarios, de interconexiones ocasionales a base de datos, del uso de ASP, LOG.

1.3 Base de datos

Una base de datos o banco de datos (en ocasiones abreviada B.D.D.) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

Existen programas denominados SGBD (sistemas gestores de bases de datos), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos SGBD, así como su utilización y administración, se estudian dentro del ámbito de la informática. (8. RAMIR, 2015)

1.3.1 Sistema de gestión de bases de datos

Los sistemas de gestión de bases de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de dato, el usuario y las aplicaciones que la utilizan.

Los hechos general de los sistemas de gestor de bases de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización. (RIPOLL, s.f.)

1.4 Microsoft SQL Server

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL. Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son *Oracle* o *MySQL*.” (Microsoft SQL Server, s.f.)

El SQL (Structured Query Language) es un estándar aceptado en productos de bases de -datos, que fue utilizado comercialmente por primera vez por la empresa Oracle en 1979.

1.5 DevExpress

DevExpress es una de las más completas suites de componentes de UI para el desarrollo en todas las plataformas de .NET como Windows Forms, ASP.NET, MVC, Silverlight y Windows 8 XAML. (García, s.f.)

1.6 Conceptos Generales

1.6.1 Historia Clínica

La Historia Clínica, es un documento que registra la experiencia médica con el paciente y representa un instrumento imprescindible para el cuidado actual o futuro, que requiere de un sistema de metodología de registro y análisis que reúna la información para análisis posteriores dentro de un contexto médico legal (Pública, Agosto 2007)

1.6.2 Factura

Las facturas, además de probar una transacción de compra o venta debe contar con ciertos datos de las partes, así como también, la clase de producto vendido y su cantidad, o bien el tipo de servicio prestado, el número y la fecha de emisión. Además, una factura, debe mostrar el precio total y unitario de la transacción, los diferentes gastos que pueden deberse a diversos conceptos y que deben abonarse al comprador, así como también, en caso de que suceda, los impuestos que la compraventa implique.

1.6.3 Modelo Cascada

El modelo en cascada es el más básico de todos, y sirve como bloque de construcción para los demás modelos de ciclo de vida. La visión del modelo cascada del desarrollo de software es muy simple; el desarrollo de software puede ser a través de una secuencia simple de fases. Cada fase tiene un conjunto de metas bien definidas, y las actividades dentro de una fase contribuyen a la satisfacción de metas de esa fase o quizás a una subsecuentica de metas de la fase. (Debitoor, s.f.)

El modelo de ciclo de vida cascada, captura algunos principios básicos:

- Planear un proyecto antes de embarcarse en él.
- Definir el comportamiento externo deseado del sistema antes de diseñar su arquitectura interna.
- Documentar los resultados de cada actividad.
- Diseñar un sistema antes de codificarlo.
- Testear un sistema después de construirlo.

2 CAPITULO II. MARCO METODOLÓGICO

2.1 Tipo de Investigación

Se establece aplicar un estudio cualitativo para lo cual se realiza una investigación basada en hechos reales, con la recolección directa de datos donde se realizan las actividades, sin manipular o utilizar variables para un enfoque estadístico.

Se asume este enfoque aprovechando la experiencia de trabajo y la disponibilidad de recursos que provee la empresa.

El enfoque de estudio se define como descriptivo, ya que es un detalle de las actividades que son realizadas dentro de un proceso operacional

2.2 Metodología Investigativa

Para el desarrollo del proyecto, se establece utilizar los siguientes métodos:

Analítico – Sintético: Se determina emplear un estudio analítico-sintético para el análisis de la fundamentación teórica para y esquematizar los contenidos del proyecto.

Empírico: Se establece el método empírico, para lo cual se realiza la recopilación de datos que provee el consultorio médico establecido en hechos existentes y lograr un enfoque descriptivo, ya que se basa en el desarrollo de las actividades realizadas dentro de los procesos empleados diariamente en el consultorio médico.

Matemático: Permitir realizar los cálculos, para la medición del proyecto. (Ingresos /salidas/cobros)

2.3 Recopilación de Información

Para realizar la recopilación de datos se determinó realizarlo por medio de entrevistas a los usuarios que ejecutan los procesos.

Método / Técnica	Objetivo	Usuario	Aplicación
Se establece el método de investigación y se realiza la reunión con el médico del consultorio que maneja los procesos.	Levantar la información detallada de los procesos que se realizan en el consultorio médico para el desarrollo de las actividades.	Medico	Conocimiento claro de los procesos

Tabla 1 Técnicas de levantamiento de información
Fuente: Autores.

2.3.1 Técnicas de Recopilación de Información

Reunión con la dueña (Doctora)

Se define el alcance de la reunión, se determinan los procesos a automatizarse y se definen a los usuarios que son responsables de los procesos.

Análisis: Al contar con una directriz de un mando alto es más fácil continuar con la entrevista a los usuarios, para que los usuarios puedan destinar tiempo de sus actividades diarias.

Entrevista con el Usuario perteneciente al área Médica: Proceso de registro de información de Cita Médica/Historial Clínico/Facturación

Proceso Manual

El área médica del consultorio se encarga de recopilar la información de la cita médica en una agenda, posteriormente escribe en un cuaderno la consulta médica para luego ser ingresa a un documento en Word llamado historia clínica.

Una vez concluida la atención médica se emite la factura solicitando los datos del cliente apellidos, nombres, dirección, teléfono, mail.

Análisis: El levantamiento con el usuario experto del proceso ayuda a entender de mejor manera la problemática y diseñar una solución automática enfocada.

Encuesta: No existe la necesidad de realizar una encuesta para determinar si la solución tiene aceptación o no ya que es aplicada a una problemática puntual que requiere una automatización.

Lo que sí se puede cuantificar en tiempo, es el proceso que actualmente toman los usuarios en ejecutar los procesos actuales. Promedio 45 minutos por consulta médica. Se espera que al automatizar la solución el proceso disminuya a 25 minutos por consulta médica.

Resultados Esperados

Al culminar el proyecto integrador se espera obtener un software que refleje toda la información del historial clínico y la facturación con los siguientes módulos.

- **Seguridades**
  - Empleados
  - Clave
  - Permiso a Roles
  - Roles
  - Asignación de Rol a Empleados
- **Gestión**
  - Pacientes
  - Proveedores
  - Clientes
  - Roles

- **Administración**
  - Departamento
  - Especialidad
  - Bodega
- **Inventario**
  - Ingreso/Egreso
  - Movimiento del producto
- **Ventas**
  - Facturación
  - Anulación de Ventas
  - Consulta de Ventas
- **Historia Clínica**
  - Cita Medica
  - Horario de Atención del Medico
  - Historial Medico
  - Consulta de Honorarios Médicos

2.4 Software

- Sistema operativo XP
- Visual Studio 2010
- Internet Google Chrome
- SQL Server 2008
- DevExpress

Todas las herramientas mencionadas anteriormente servirán para el buen diseño y posterior desarrollo del sistema de gestión de historias clínicas y facturación del consultorio clínico Salud & Vida.

2.5 Estudio de la Metodología y Herramientas.

2.5.1 Modelo de ciclo de vida

El ciclo de vida que se aplicó en el proyecto (Gestión del historial clínico y facturación) es el modelo en cascada; porque permite la posibilidad de hacer iteraciones, es decir, durante las modificaciones que se hacen en el mantenimiento se puede ver por ejemplo la necesidad de cambiar algo en el diseño, lo cual significa que se harán los cambios necesarios en la codificación y se tendrán que realizar de nuevo las pruebas, es decir, si se tiene que volver a una de las etapas anteriores al mantenimiento hay que recorrer de nuevo el resto de las etapas. (Navas, s.f.)

El modelo que esta aplicado al proyecto se puede apreciar en el Grafico 1.

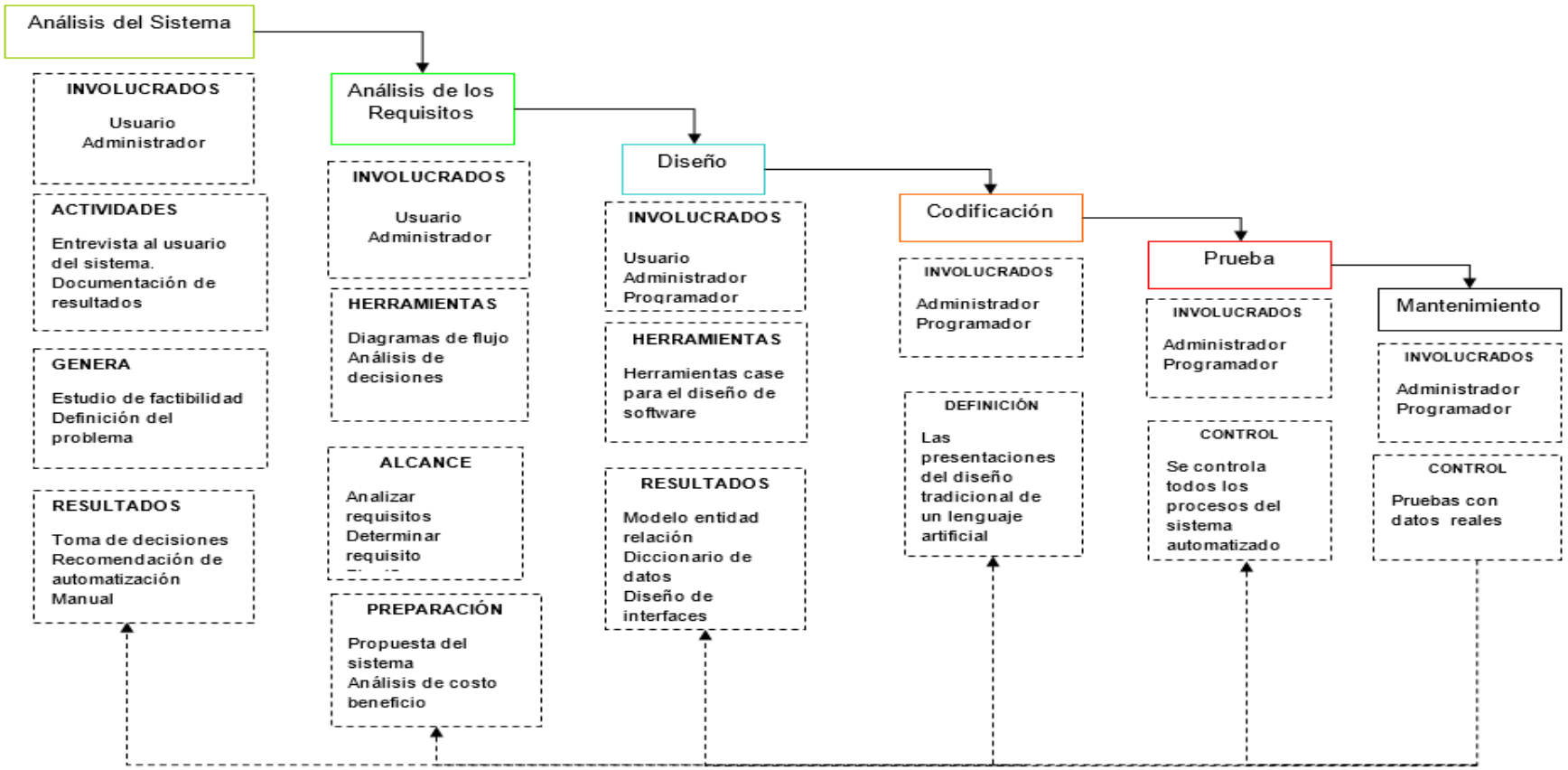


Gráfico 1 Modelo Ciclo de Vida
Fuente: Autores

2.5.2 Selección de las herramientas a utilizar

Se va seleccionar las herramientas más adecuadas para el desarrollo del sistema de gestión del historial clínico y facturación Las herramientas a utilizar son: Entorno integrado de desarrollo Visual Studio.net 2010, Lenguaje de programación C#.Net, Gestor de base de datos SQL Server 2008.

2.5.3 Lenguajes de Programación

Cuadro comparativo para el lenguaje de programación que se necesitará.

No	FUNCTION	Java	PowerBuilder	Visual.Net	C#
1	Conexión a la BDD	X	X	X	X
2	Tipos de Datos	X		X	X
4	Menor Costo	X			X
5	Disponibilidad		X	X	X
6	Drivers		X		
7	Entorno Gráfico	X			X
8	Mayor Competencia				X
9	Tecnología	X	X	X	X
10	Herramientas				X
	SELECCIÓN				X

Tabla 2 Lenguaje de programación
Fuente: Autores.

Por las múltiples ventajas que proporciona la conexión a SQL Server, permite utilizar distintos tipos de datos y permite la programación orientada a objetos, se va utilizar el lenguaje de programación C#

2.5.4 Plataformas a utilizar

Se ha tomado la decisión de utilizar la plataforma de Windows XP ya que es compatible con el lenguaje de programación y el gestor de base de datos a utilizar y no da ningún problema de compilación para el desarrollo del sistema.

2.5.5 Base de Datos

No	FUNCIÓN	My Sql	Sql Server 2008	Oracle
1	Facilidad de instalación y configura	X	X	X
2	Drivers		X	
3	Disponibilidad		X	
4	Estabilidad de Almacenamiento	X	X	
5	Modelo Relacional		X	X
6	Diseño BD		X	
	SELECCIÓN		X	

Tabla 3 Gestor de base de datos
Fuente: Autores.

El sistema gestor para la base de datos que se va utilizar para el sistema es SQL server 2008 ya que compatible con el lenguaje de programación y la plataforma que se va utilizar y además es uno de los gestores de base de datos más estables que existen.

2.6 Herramientas Metodológicas

Herramientas metodológicas seleccionadas para el análisis y diseño del sistema.

2.6.1 Power Designer

Se utilizó para poder realizar el modelo físico y lógico los cuales permitirán el diseño de la base de datos del sistema.

2.6.2 BPMN 2.0

Va permitir la creación de los diagramas de flujo del sistema, los cuales van a permitir saber cuáles van a ser los procesos que va a tener el sistema.

2.6.3 Pencil Project

Es una herramienta que se utilizó para crear prototipos GUI.

2.6.4 StarUML

Es una herramienta que se utilizó para el modelamiento de software.

2.6.5 DevExpress

DevExpress es una de las más completas suites de componentes de UI para el desarrollo en todas las plataformas de .NET como Windows Forms, ASP.NET, MVC, Silverlight y Windows 8 XAML. (García, s.f.)

2.7 Análisis

2.7.1 Levantamiento de información

Después de haber aplicado la entrevista se logró la siguiente información importante.

- El consultorio médico no cuenta con un archivo físico del historial clínico de sus pacientes debido al alto costo de papelería y mantenimiento del mismo.
- No existe una agenda médica para llevar el control de las citas médicas con los pacientes.
- El consultorio médico lleva la información del paciente en un documento físico (Cuaderno) que posteriormente es transcrito en un documentó de Word.
- La facturación se la lleva de manera física.

2.7.2 Análisis de resultados

El proceso de historial clínico y facturación no se lleva de forma correcta porque la información recabada tanto del paciente como de la facturación son vulnerables, este problema conlleva a no tener un adecuado historial de los pacientes que son atendidos por diferentes patologías.

2.7.3 Análisis Integral

Es importante implementar un sistema de gestión de historial clínico y facturación automatizado puesto que se presentan varios inconvenientes al gestionar el control médico del paciente y proceso de facturación.

Definir de forma clara y precisa los parámetros que se desea que contenga el software, para un correcto entendimiento y fácil manipulación de los usuarios, por tanto debe contener el proceso de atención médica y facturación que le permita manejar de mejor manera el consultorio.

2.7.4 Descripción de la Situación Inicial

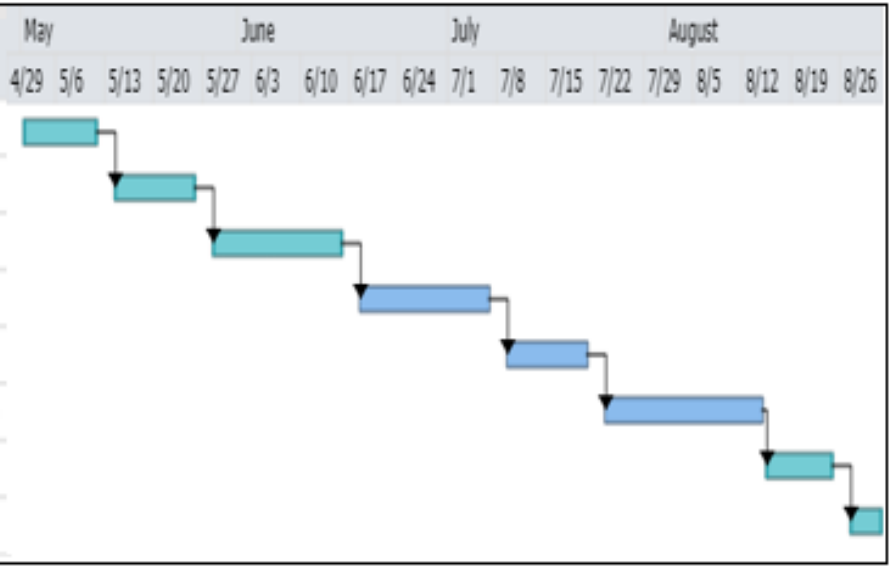
La información que se recaba de los pacientes es de difícil manejo y a la vez vulnerable, por no estar dicha información almacenada en una base de datos, este problema conlleva a no poseer un adecuado control estadístico de los pacientes que son atendidos por diferentes patologías, y que a su vez dificulte brindar una atención preventiva.

Al término de la atención del paciente se emite una factura realizada manualmente, lo cual ocasiona diversos errores al efectuar la suma de valores y totalizar la misma, por estos inconvenientes surge la necesidad de crear un software que permita optimizar y solucionar las necesidades que demanda la empresa de salud.

Teniendo en cuenta todos los problemas mencionados anteriormente, es necesario desarrollar un sistema que automatice la historia clínica y facturación que permita brindar un mejor servicio al cliente.

2.8 Cronograma de las Tareas a Realizar

Name	Duration	Start	Finish
Recopilación de Información	9 days	Tue 5/1/18 9:00 AM	Fri 5/11/18 7:00 PM
Elaboración del diseño del plan	10 days	Mon 5/14/18 9:00 AM	Fri 5/25/18 7:00 PM
Elaboración del esquema de BD	15 days	Mon 5/28/18 9:00 AM	Fri 6/15/18 7:00 PM
Diseño de Interfaces	15 days	Mon 6/18/18 9:00 AM	Fri 7/6/18 7:00 PM
Construcción del Módulo de Mantenimiento	10 days	Mon 7/9/18 9:00 AM	Fri 7/20/18 7:00 PM
Construcción de Interfaces del Proceso	17 days	Mon 7/23/18 9:00 AM	Tue 8/14/18 7:00 PM
Elaboración de Reportes	8 days	Wed 8/15/18 9:00 AM	Fri 8/24/18 7:00 PM
Implementación	5 days	Mon 8/27/18 9:00 AM	Fri 8/31/18 7:00 PM



3 CAPÍTULO III. PROPUESTA

3.1 Diagramas de Procesos

3.1.1 Proceso manual actual

El registro de pacientes, citas médicas, historial clínico y facturación (Tiempo estimado actual 45 minutos)

En el Grafico 2 se puede observar el proceso manual del consultorio salud y vida que toma un tiempo de atención al paciente entre (40 – 50 minutos) teniendo una media de 45 minutos.

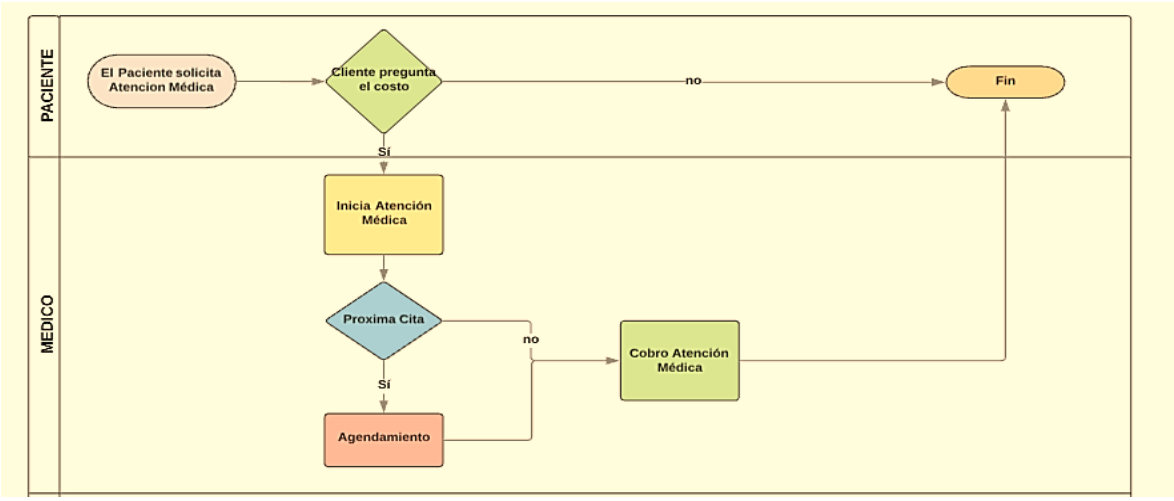


Gráfico 2 Proceso actual, no automatizado
Fuente: Autores

3.1.2 Proceso general propuesto.

Se visualiza el proceso automatizado de registro de clientes, citas médicas, historial clínico y facturación. El diagrama fue verificado con el medico del consultorio. (Tiempo estimado 25 minutos)

En el Grafico 3 se puede observar el proceso automatizado planteado para reducir el tiempo de espera del paciente que acude al consultorio Salud & Vida a un tiempo aproximado de 25 minutos.

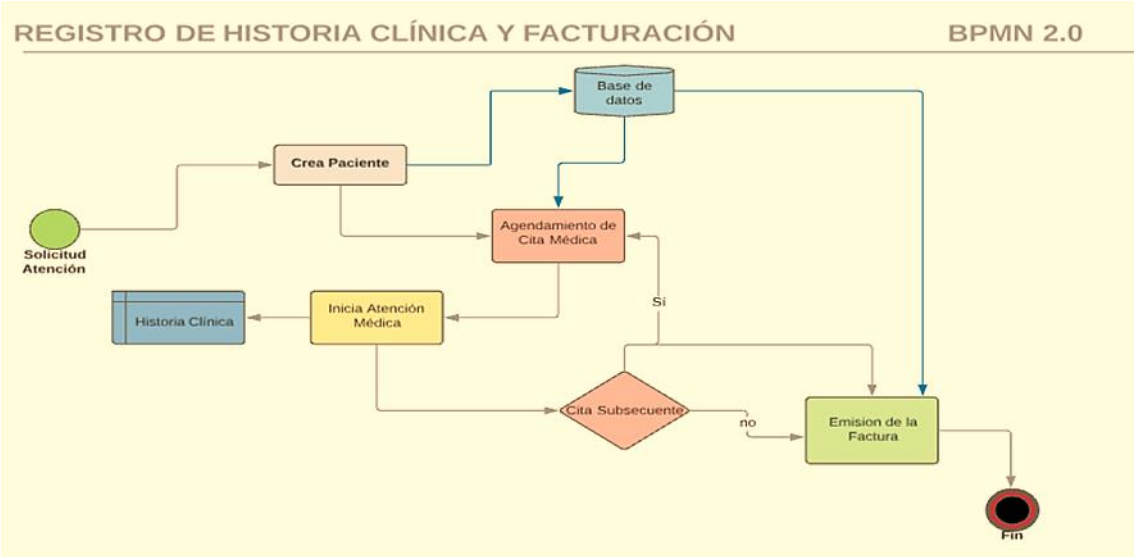


Gráfico 3 Proceso propuesto automatizado
Fuente: Autores.

3.2 Especificación de Requerimientos

3.2.1 Ámbito del Software

La solución a implementar es un sistema desarrollad en Visual Studio 2010, SQL Server 2008 y UI DevExpress, como gestión de Historial Clínico y Facturación.

Funcionalidad esperada:

- El sistema contara con el módulo de seguridades que permitirá gestionar la creación de usuarios y asignación de roles de cada empleado.
- El sistema permitirá la creación de clientes y pacientes, para lo cual solicitara ingresar los datos generales y específicos.
- El sistema llevara un registro de ingreso y egreso de productos, para el control ‘y existencia de los mismos.

- El sistema permitirá ingresar los horarios de atención del médico, conforme a su disponibilidad.
- Una vez registrado los datos del paciente, el sistema accederá el agendamiento de citas médicas, validando los horarios disponibles para la atención.
- El usuario contara con el módulo de historial clínico, en el cual permitirá registrara y consultar el historial clínico del paciente.
- El usuario contara con una opción de consultas de ventas, movimientos de productos y honorarios médicos.

Beneficio

- Ahorro de tiempo operativo.
- Centralización de la información.
- Gestión de consulta por reportes.

Beneficio

- Minimizar el riesgo de errores del ingreso manual de información lo que puede ocasionar dificultades en próximas consultas médicas del paciente.
- Reducir el tiempo empleado en cada proceso de ingreso de información.
- Mantener un control centralizado de la información de los pacientes.

3.2.2 Funciones del Producto

- Administración de pacientes, proveedores, clientes, roles.
- Registro de ingreso/egreso de productos
- Creación de la cita médica.
- -Creación del Historial clínico
- Registro de ventas

Historias de Usuarios: Una vez realizadas las entrevistas a través de reuniones con los usuarios claves se detallan las historias de usuarios levantadas.

HISTORIA DE USUARIO	
Número: 1	**Nombre:** Dirección Medica
Usuario: Medico	**Riesgo en Desarrollo:** Alta
Prioridad en negocio: Alta	**Iteración asignada:** 1
Descripción: Modo de inicio de sesión al sistema. - Los usuarios del sistema inician sesión. - Adicional al punto anterior es necesario que el usuario se encuentre creado dentro de la administración de usuarios del Sistema. - El usuario debe tener un rol definido para el acceso a las distintas opciones que provee el sistema.	
Observación: Los usuarios definidos para el proceso ya cuentan con una cuenta y están creados en el sistema, para los nuevos empleados necesarios crear un rol y asignar al usuario.	

Tabla 4 Historia de Usuario 1 Inicio de Sesión
Fuente: Autores

HISTORIA DE USUARIO	
Número: 2	**Nombre:** Médico Especialista
Usuario: Médico Especialista	**Riesgo en Desarrollo:** Alta
Prioridad en negocio: Alta	**Iteración asignada:** 1
Descripción: Detalle del proceso de registro de Historia Clínica. - El usuario accede al sistema mediante inicio de sesión - El usuario debe crear la cita para que le aparezca en la agenda médica. - El usuario ingresa al formulario Historial y selecciona el paciente que va a ser atendido. - El usuario debe registrar el motivo de consulta, enfermedad actual, diagnósticos acorde al CIE-10 y plan de tratamiento. - El usuario imprime la receta médica en papel química de 2 hojas el original es para el cliente y la copia se queda como respaldo interno del consultorio.	
Observación: El médico debe tener en cuenta las alergias del paciente para la administración y/o prescripción de medicamentos.	

Tabla 5 Historia de Usuario 2 Proceso de Registro de Historia Clínica
Fuente: Autores

HISTORIA DE USUARIO	
Número: 3	**Nombre:** Médico Especialista
Usuario: Médico Especialista	**Riesgo en Desarrollo:** Alta
Prioridad en negocio: Alta	**Iteración asignada:** 1
Descripción: Detalle del proceso de registro de Factura. - El usuario accede al sistema mediante inicio de sesión	

- El usuario debe seleccionar el tipo de factura (paciente, cliente), si el paciente desea que la factura se emita con sus datos debe seleccionar **PACIENTE** y la opción **TERCEROS** es para menores de edad que requieren se emita con los datos del titular. - El usuario selecciona las medicinas e insumos que se utilizaron para su recuperación y/o tratamiento. - El usuario debe selecciona la consulta médica y asignar el nombre del médico que le atendió para que se registre sus honorarios médicos. - El usuario imprime la factura en papel química de 2 hojas el original es para el cliente y la copia se queda como respaldo interno del consultorio.
Observación: El consultorio médico tiene un alto consumo de papel por las impresiones realizadas.

Tabla 6 Historia de Usuario 3 Registro de Facturación
Fuente: Autores

3.2.3 Características de los Usuarios del Sistema

En la siguiente tabla se muestran las características de los usuarios categorizados por perfiles.

Nombre del Rol	Tipo de Usuario	Área Funcional	Actividad
Administrador	Administrador del sistema de Gestión Historial Clínico y Facturación	Seguridades	Creación de usuarios y asignación de roles. Soporte.
Medico	Medico	Salud	Creación de pacientes, citas médicas e historial clínico.
Técnico	Técnico	Administración	Consultar al sistema. Registrar materiales. Ver reportes.

Tabla 7 Perfiles de usuario
Fuente: Autores

3.2.4 Restricciones

Algunas de las restricciones que tiene el sistema son:

- El sistema no va permitir emitir facturas electrónicas.
- El sistema solo puede ser utilizado Microsoft Visual Studio 2010.

- El sistema constara con 3 reportes (ventas, honorarios médicos y moviente de productos)

3.2.5 Requisitos

Los requisitos que el usuario solicita en el desarrollo del sistema están plasmados en las listas de requerimientos funcionales y no funcionales mostrados a continuación

FUNCIONALES

RF01: El usuario podrá acceder a la opción de citas médicas del rol que se creara para dicho fin.

RF02: El usuario podrá realizar consulta de ventas y posteriormente exportar a Excel.

RF03: El usuario podrá modificar la información del producto y/o servicio del rol que se creara para dicho fin.

RF04: El usuario podrá consultar los honorarios médicos

NO FUNCIONALES

RNF01: Usabilidad.- Las interfaces creadas mantienen el diseño estándar de uso familiar para el usuario.

RNF02: Plataforma.- El software se encuentra instalado en plataformas Windows.

RNF03: Rendimiento.- La arquitectura que se encuentra establecida está cuantificada para grandes manejos de información.

RNF04: Desempeño.- Las opciones creadas no presentaran problemas para su manejo e implementación.

4 CAPÍTULO IV. IMPLEMENTACIÓN

4.1 Diseño General

Para la ejecución del proyecto se adoptó la metodología de desarrollo Ágil XP conocida por sus siglas XP (**eXtreme Programming/ Programación extrema**). Se utilizaron las siguientes herramientas orientadas a Objetos: Plan de entregas, tarjetas CRC y pruebas de aceptación.

4.1.1 Plan de entrega

N°	Descripción	Fecha inicio	Fecha fin	Observaciones
1	Exploración: - Planteamiento del problema. - Historia de Usuario	23/04/2018	11/05/2018	Reunión con la Dra. Katya Caisa propietaria del Consultorio Salud & Vida. Se establece el problema a solventar. Se realiza el levantamiento de información. Se realiza el levantamiento de todas las actividades.
2	Planificación: - Definición del alcance. - Definición de requerimientos - Definición de los recursos - Cronograma	14/05/2018	18/05/2018	Reunión con la Dra. Katya Caisa propietaria del Consultorio Salud & Vida. Planteamiento de la solución, alcance. Definición de recursos (usuarios claves). Reunión con los recursos asignados: Médicos Generales Definición de la infraestructura. Definición de la participación de los usuarios en las etapas del proyecto.
3	Iteraciones: - Desarrollo - Pruebas	21/05/2018	20/07/2018	Codificación del desarrollo. Pruebas con los usuarios Pruebas funcionales Pruebas no funcionales
4	Producción: - Implementación en el ambiente de Producción.	23/07/2018	03/08/2018	Capacitación a usuarios. Paso del producto desarrollado y probado al ambiente de producción.
5	Mantenimiento	06/08/2018	17/08/2018	Soporte

6	Fin: Finalización del Proyecto	20/08/2018	24/08/2018	Cierre del proyecto.

Tabla 8 Plan de entrega
Fuente: Autores

4.1.2 Tarjetas CRC

DelInicioSesión	
Responsabilidades	Colaboradores:
AccesoSistema	Usuario

Tarjeta CRC 1 Inicio de Sesión

DelAdministrador	
Responsabilidades:	Colaboradores:
CreaEmpleado CreaRol AsignaRol CreaProductos/Servicios IngresaProductos/Servicios CreaHorarioAtencion	Usuario

Tarjeta CRC 2 Administrador

DelRegistroCitaMedica	
Responsabilidades:	Colaboradores:
CrearPaciente CreaCitaMedica InsertaCitaMedica ActualizaCitaMedica	Usuario

Tarjeta CRC 3 Registro de Cita Médica

DelRegistroHistoriaClinica	
Responsabilidades:	Colaboradores:
CreaHistorialClinico InsertaConsultaMedica InsertaEvolucionMedica InsertaRecetaMedica	Usuario

Tarjeta CRC 4 Registro de Historia Clínica

DelRegistroFacturacion	
Responsabilidades:	Colaboradores:
CrearCliente InsertaEncabezadoVenta InsertaDetalleVenta	Usuario

Tarjeta CRC 5 Registro de Facturación

Diseño Detallado

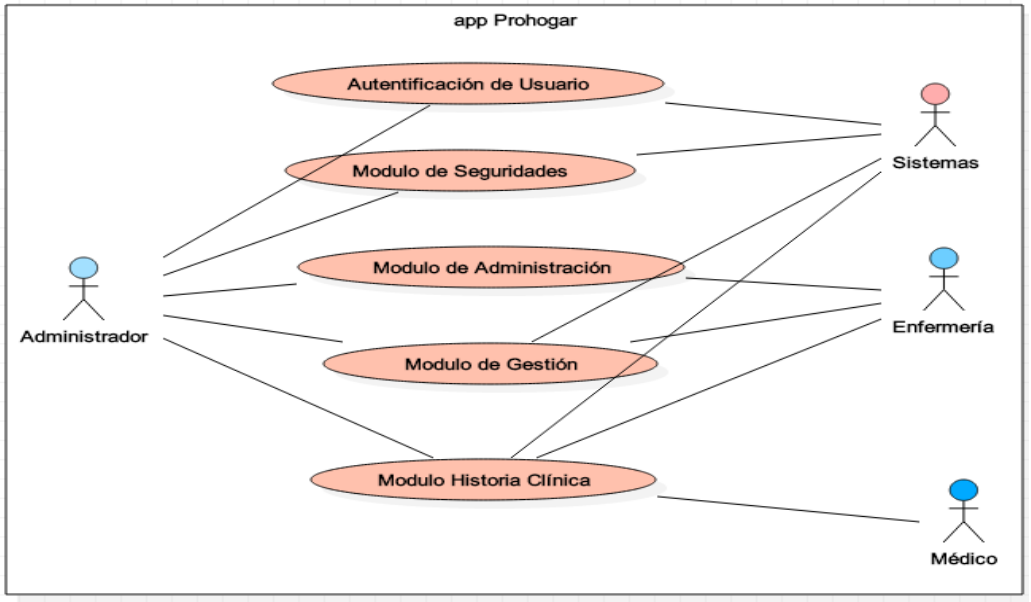


Gráfico 4 Diagrama De Caso De Uso General
Fuente: Autores

DIAGRAMA DE CASO DE USOS GENERAL	SOFTWARE PARA LA GESTIÓN DEL HISTORIAL CLÍNICO Y FACTURACIÓN.
ACTOR	**FUNCIÓN**
Sistemas:	Es el encargado de gestionar y brindar soporte técnico al operador que lo requiera para el correcto funcionamiento del sistema.
Administrador:	Es el encargado de la seguridad del sistema controlando la creación y manejo de los perfiles de acceso al sistema.
Enfermería:	Es el encargado de gestionar la cita médica y generar la factura de venta.
Médico:	Es el encargado de registrar los datos del paciente para realizar la historia clínica.

Tabla 9 Diagrama Análisis de Uso General
Fuente: Autores'

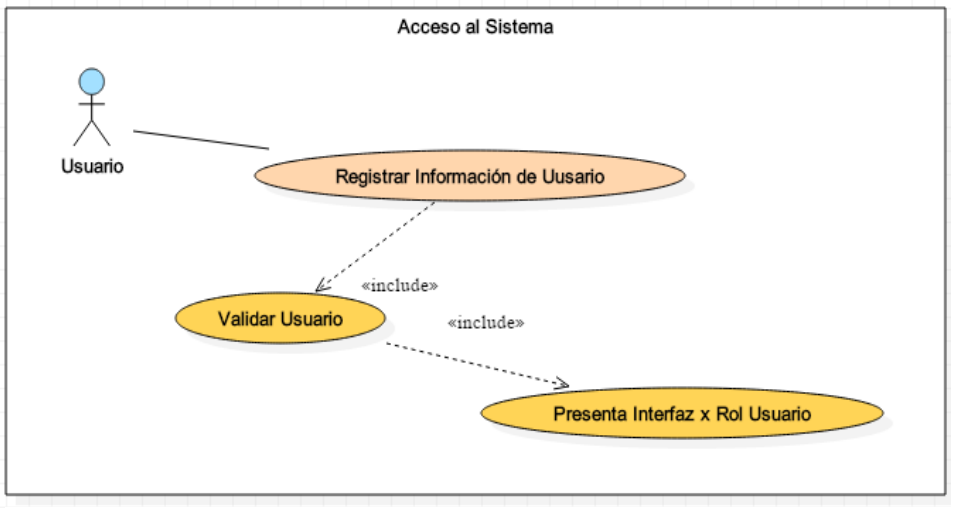


Gráfico 5 Caso De Uso: Acceso Al Sistema
Fuente: Autores

ESCENARIO	
NOMBRE CASO DE USO:	**ACCESO AL SISTEMA**
Descripción:	El empleado ingresa usuario y contraseña al sistema posteriormente es validado y permite el acceso a su perfil
Pre-condiciones:	El usuario esta creado en el sistema
Pos-condiciones:	El usuario accede al sistema según su perfil
FLUJO NORMAL	
ACTOR	**USUARIO**
1.- Registrar el usuario y contraseña	
2.- Presionar Aceptar	
	3.- Conexión BD **Conexión BD()**
	4.-Sistema valida usuario en la BD **Validar Empleado()**
	5.-Cargar perfil y presentar la **Cargar Perfil Empleado()** Pantalla Principal
FLUJO ALTERNATIVO	
F1	Cargar pantalla recuperación contraseña
F2	El usuario presiona salir y sistema será cerrado
F3	Mensaje de error : ”Fallo Conexión con BD” ,
F4	Mensaje de error: ”Usuario no encontrado”

Tabla 10 Diagrama Análisis Acceso al Sistema
Fuente: Autores

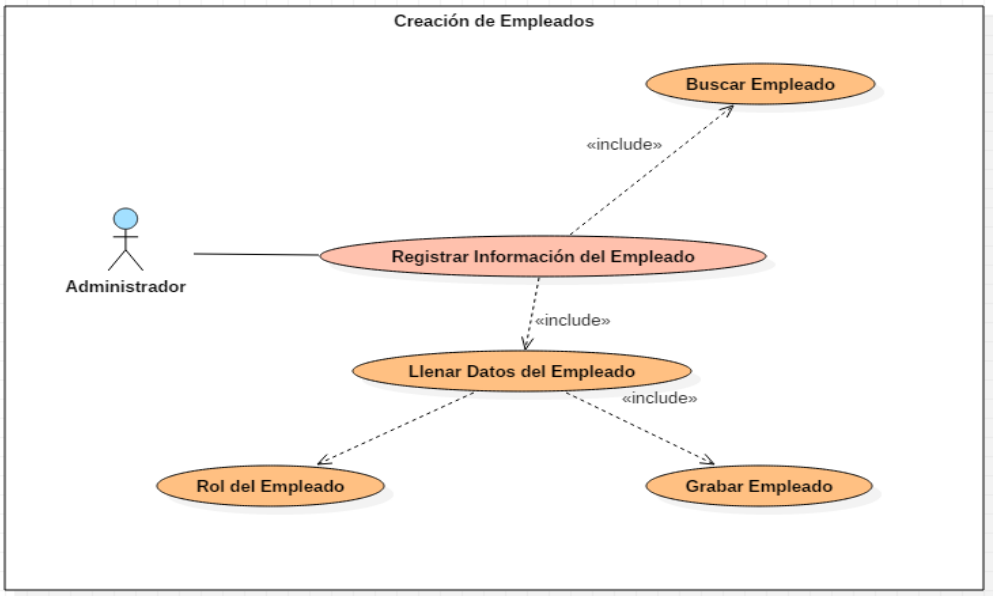


Gráfico 6 Caso De Uso: Creación De Empleados
Fuente: Autores

ESCENARIO	
NOMBRE CASO DE USO:	**CREACIÓN DE EMPLEADOS**
Descripción:	El usuario ingresa nombres, apellidos, dirección, teléfono, nro. Identificación, usuario, contraseña, mail, selecciona el género, estado civil, departamento, una vez ingresada toda la información debe presiona grabar para que se guarde el registro del empleado en la BD.
Pre-condiciones:	El empleado esta creado en el sistema
Pos-condiciones:	Empleado creado
FLUJO NORMAL	
ACTOR	**ADMINISTRADOR**
1.- Ingresar datos del empleado	
	2.- Realiza búsqueda en BD **Buscar Empleado()**
3.- Ingresa nombre de usuario, contraseña	
	4.- El sistema valida contraseña **Valida Contraseña()**
5.- Presionar Grabar	
	6.-Inserta registros de Empleado en BD **Grabar Empleado()**
FLUJO ALTERNATIVO	
F1	Mensaje de error: “ Ingrese información correcta”

F2	Mensaje de error: “ Registro no Grabado”

Tabla 11 Diagrama Análisis Creación Empleados
Fuente: Autores

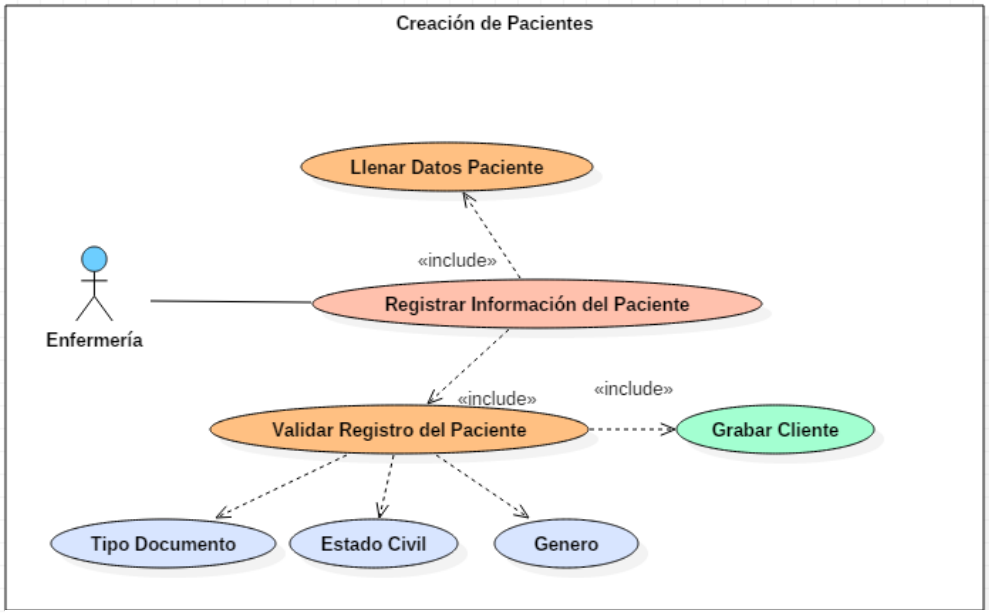


Gráfico 7 Caso De Uso: Creación De Pacientes
Fuente: Autores

ESCENARIO	
NOMBRE CASO DE USO:	**CREACIÓN DE PACIENTES**
Descripción:	El usuario ingresa datos del cliente y valida cada registro para posteriormente ser creado en el sistema
Pre-condiciones:	El paciente existente en el sistema o nuevo
Pos-condiciones:	Paciente creado
FLUJO NORMAL	
ACTOR	**ENFERMERA**
1.- Ingresa nombres, apellidos, cédula dirección, teléfono	
	2.- El sistema valida cada registro **Validar Paciente()**
3.- Presionar Grabar	
	4.-Inserta registros de empleado en BD **Grabar Paciente ()**
FLUJO ALTERNATIVO	

F1	Mensaje de error: “ Registro no Valido”
F2	Mensaje de error: “ Registro no Grabado”

Tabla 12 Diagrama Análisis Creación Paciente
Fuente: Autores

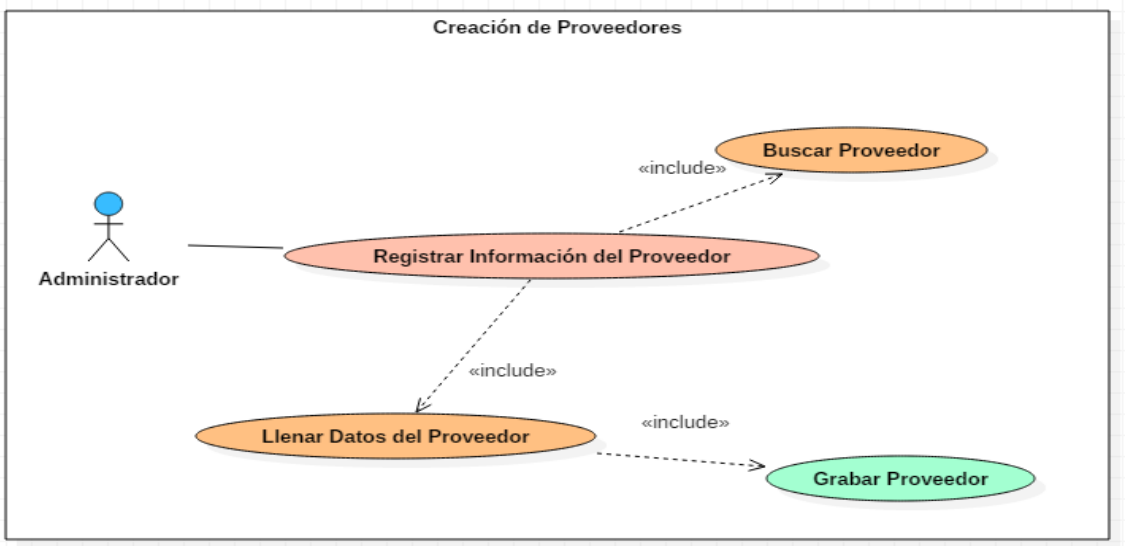


Gráfico 8 Caso De Uso: Creación De Proveedores
Fuente: Autores

ESCENARIO	
NOMBRE CASO DE USO:	**CREACIÓN DE PROVEEDORES**
Descripción:	El usuario ingresa datos del proveedor y valida cada registro para posteriormente ser creado en el sistema
Pre-condiciones:	El proveedor existente en el sistema
Pos-condiciones:	Proveedor creado
FLUJO NORMAL	
ACTOR	**ADMINISTRADOR**
1.- Ingresa nombres, apellidos, cédula o ruc, dirección, teléfono	
2.-Seleccionar Tipo Identificación	
	3.- El sistema valida cada registro **Validar Proveedor()**
4.- Presionar Grabar	
	5.-Inserta registros de empleado en BD **Grabar Proveedor()**

FLUJO ALTERNATIVO	
F1	Mensaje de error: “ Registro incorrecto”
F2	Mensaje de error: “ Registro no Grabado”

Tabla 13 Diagrama Análisis Creación Proveedor
Fuente: Autores

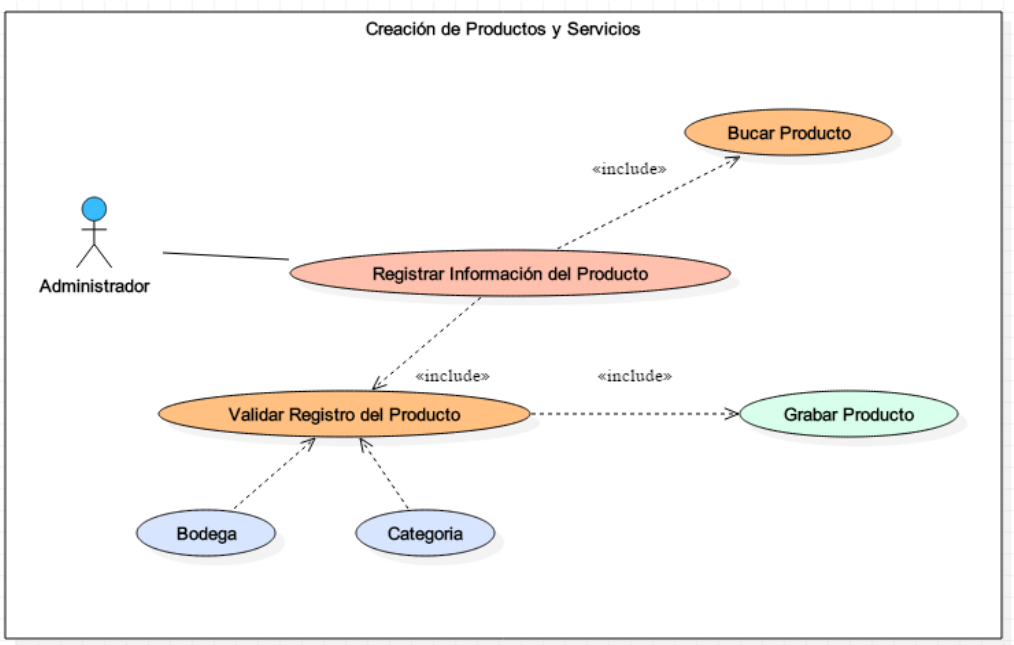


Gráfico 9 Caso De Uso: Creación De Productos Y Servicios
Fuente: Autores

ESCENARIO	
NOMBRE CASO DE USO:	**CREACIÓN DE PRODUCTOS**
Descripción:	El usuario selecciona categoría y registra los datos del producto y el sistema valida cada registro para posteriormente ser creado.
Pre-condiciones:	El producto existente en el sistema
Pos-condiciones:	Producto creado
FLUJO NORMAL	
ACTOR	**ADMINISTRADOR**
1.- Seleccionar Categoría	
	2.- Sistema busca Categoría **Recuperar Categoría()**
3.- Ingresa datos del producto	
4.- Presionar Grabar	

	5.-Inserta registros de producto en BD **Grabar Producto()**
FLUJO ALTERNATIVO	
F1	Mensaje de error: “ tipo producto no existe”
F2	Mensaje de error: “ Registro no Grabado”

Tabla 14 Diagrama Análisis Creación Producto / Servicio
Fuente: Autores

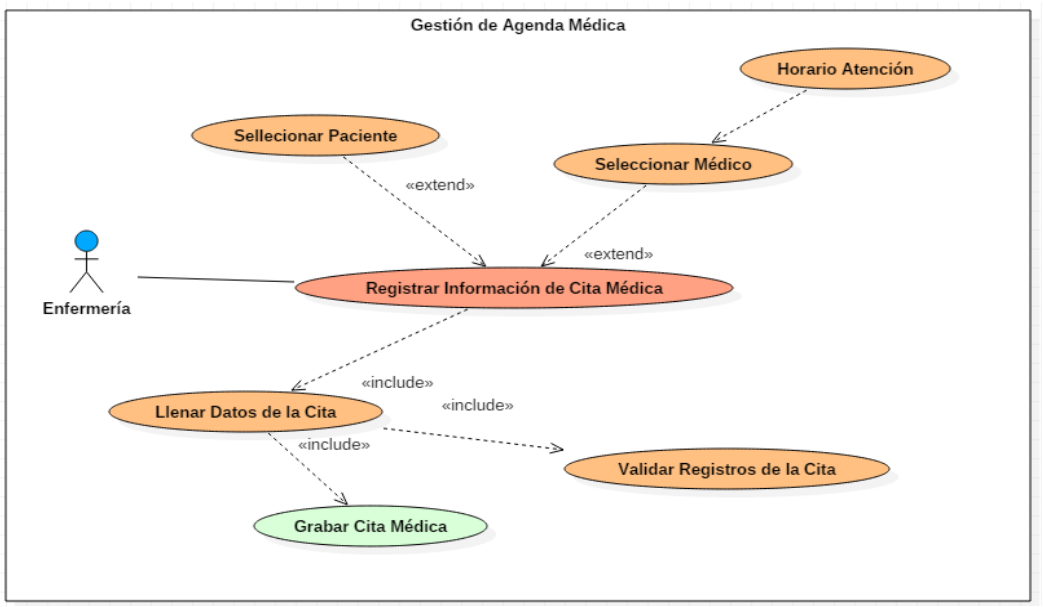


Gráfico 10 Caso De Uso: Cita Médica
Fuente: Autores

ESCENARIO	
NOMBRE CASO DE USO:	**CITA MÉDICA**
Descripción:	El usuario registra la cita médica acorde al horario de atención del médico para reservar su atención en la agenda del médico.
Pre-condiciones:	El paciente ya tiene reservada la cita en el sistema
Pos-condiciones:	Cita médica creada
FLUJO NORMAL	
ACTOR	**ENFERMERA**
1.- Seleccionar Paciente	
2.-Seleccionar Médico	
	3.- El sistema valida cada registro **Validar Cita()**
4.- Presionar Grabar	
	5.-Inserta registros de cita médica en BD

	Grabar Cita()
FLUJO ALTERNATIVO	
F1	Mensaje de error: “ Registro incorrecto”
F2	Mensaje de error: “ Registro no Grabado”

Tabla 15 Diagrama Análisis Creación Cita Médica
Fuente: Autores

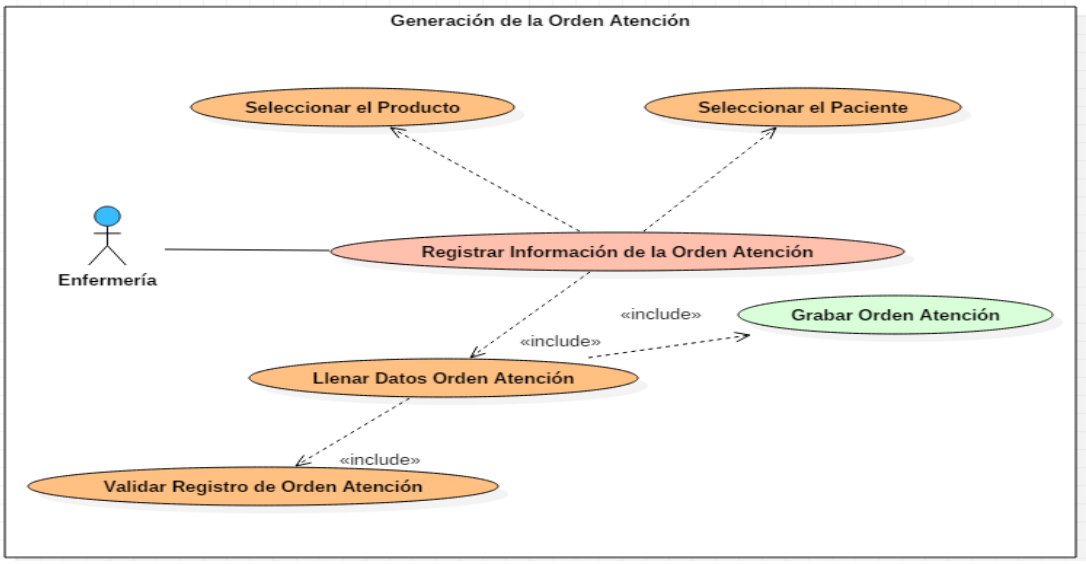


Gráfico 11 Caso De Uso: Orden De Atención
Fuente: Autores

ESCENARIO	
NOMBRE CASO DE USO:	**CREACIÓN DE ORDEN DE ATENCIÓN**
Descripción:	El usuario genera el pedido de consumos.
Pre-condiciones:	Admisión
Pos-condiciones:	Generar la orden de atención
FLUJO NORMAL	
ACTOR	**ENFERMERA**
1.- Seleccionar Paciente	
2.- Nuevo Pedido	
	3- Conexión BD **Conexión BD()**
	4- Sistema busca cliente **Recuperar Paciente()**
	5.- Sistema busca producto **Recuperar Producto()**
6.- Presionar Grabar	
	7.-Inserta registros de encabezado en BD

	Grabar Encabezado()
	8.-Inserta registros de detalle en BD **Grabar Detalle()**
FLUJO ALTERNATIVO	
F1	Mensaje de error: “ No existe Admisión”
F2	Mensaje de error: “ Registro no Grabado”

Tabla 16 Diagrama Análisis Creación Orden de Atención
Fuente: Autores

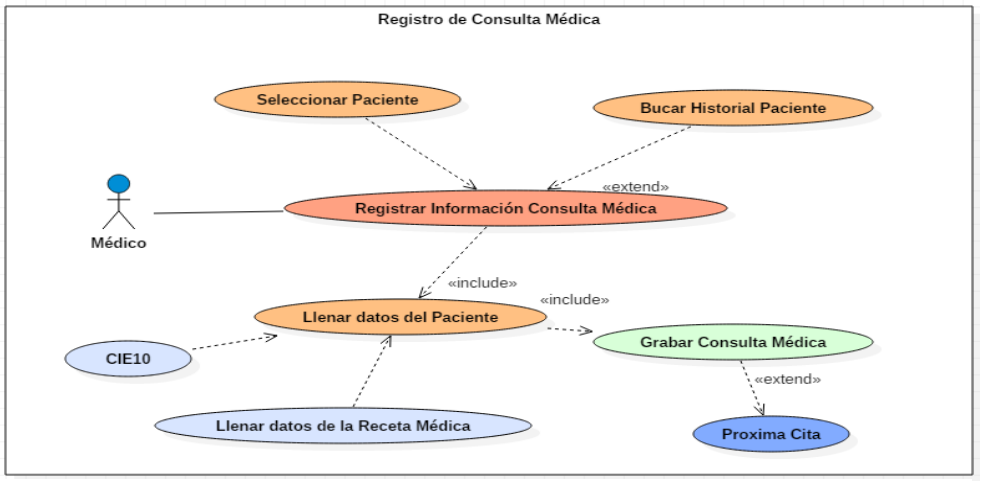


Gráfico 12 Caso De Uso: Consulta Médica
Fuente: Autores

ESCENARIO	
NOMBRE CASO DE USO:	**CONSULTA MÉDICA**
Descripción:	El Médico registra la información del estado de salud del paciente.
Pre-condiciones:	Historial Clínico
Pos-condiciones:	Generar la consulta de atención
FLUJO NORMAL	
ACTOR	**MÉDICO**
1.- Seleccionar Paciente	
2.- Ingresar datos para el registro de atención	
	3-Conexión BD **Conexión BD()**
4.- Presionar Grabar	
	5.- Inserta registros de consulta médica en BD **Grabar Consulta()**

	6.- Inserta registros de CIE en BD **Grabar CIE()**
	7.- Inserta receta médica en BD **Grabar Receta()**
FLUJO ALTERNATIVO	
F1	Mensaje de error: “ Registro no Grabado”

Tabla 17 Diagrama Análisis Creación Consulta Médica
Fuente: Autores

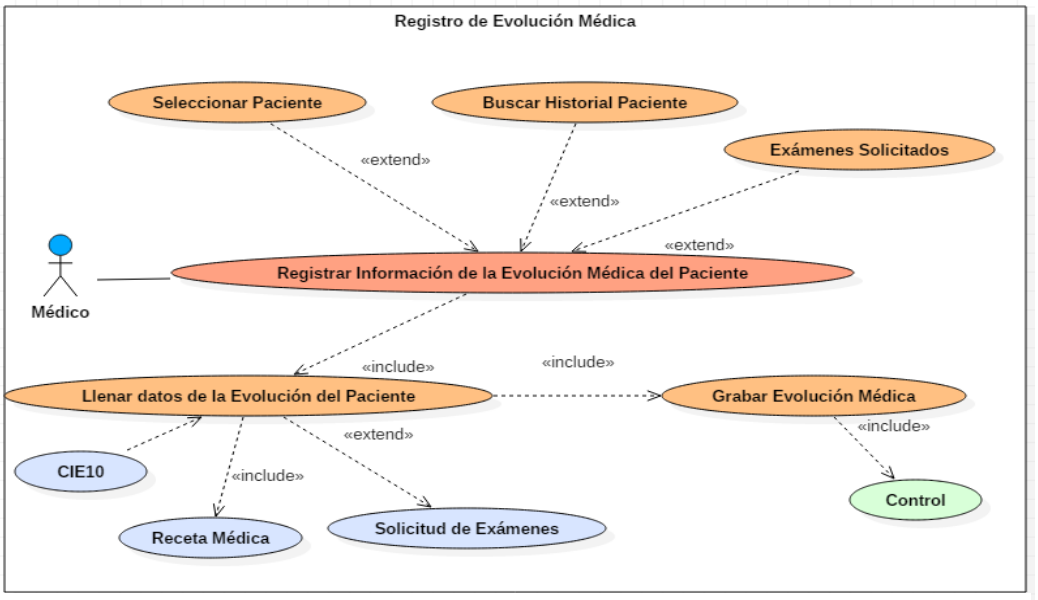


Gráfico 13 Caso De Uso: Evolución Médica
Fuente: Autores’

ESCENARIO	
NOMBRE CASO DE USO:	**EVOLUCIÓN MÉDICA**
Descripción:	El Médico registra la información de la evolución de salud del paciente.
Pre-condiciones:	Historial Clínico
Pos-condiciones:	Generar la nota de evolución
FLUJO NORMAL	
ACTOR	**MÉDICO**
1.- Seleccionar Paciente	
2.- Ingresar datos para el registro de nota de evolución	
	3-Conexión BD **Conexión BD()**

4.- Presionar Grabar	
	5.- Inserta registros de nota de evolución en BD **Grabar Evolución()**
	6.- Inserta registros de CIE en BD **Grabar CIE()**
	7.- Inserta receta médica en BD **Grabar Receta()**
FLUJO ALTERNATIVO	
F1	Mensaje de error: " No existe paciente"
F2	Mensaje de error: " Registro no Grabado"

Tabla 18 Diagrama Análisis Creación Evolución Médica
Fuente: Autores

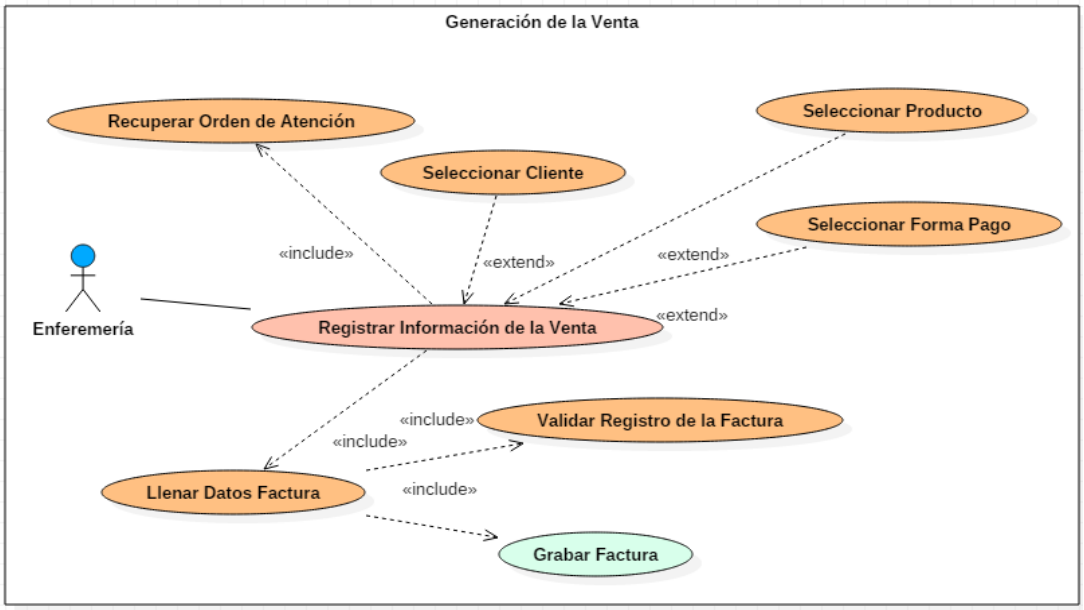


Gráfico 14 Caso De Uso: Creación De Factura
Fuente: Autores

ESCENARIO	
NOMBRE CASO DE USO:	**CREACIÓN DE FACTURA**
Descripción:	El usuario lleva el control de ventas, creación de clientes y genera la factura
Pre-condiciones:	La factura es validada
Pues-condiciones:	Generar la factura
FLUJO NORMAL	
ACTOR	**ENFERMERÍA**
1.- Seleccionar cliente	

2.- Seleccionar forma pago	
3.- Nuevo cliente	
	4- Conexión BD **Conexión BD()**
	5- Sistema busca cliente **Recuperar Cliente()**
	6.- Sistema busca producto **Recuperar Producto()**
	7.- Sistema busca forma pago **Buscar Forma Pago ()**
	8.- Insertar registro nuevo cliente BD **Grabar Cliente()**
9.- Presionar Grabar	
	10.-Inserta registros de encabezado en BD **Grabar Encabezado()**
	11.-Inserta registros de detalle en BD **Grabar Detalle()**
FLUJO ALTERNATIVO	
F1	Mensaje de error: " Cliente no existe" Contactar Administrador
F2	Mensaje de error: " Registro no Grabado" Contactar Administrador

Tabla 19 Diagrama Análisis Creación de la Factura
Fuente: Autores

4.2 Esquema de la Base de Datos

La solución planteada esta soportada sobre un modelo relacional de Bases de Datos, se muestra el modelo físico de las estructuras utilizadas.

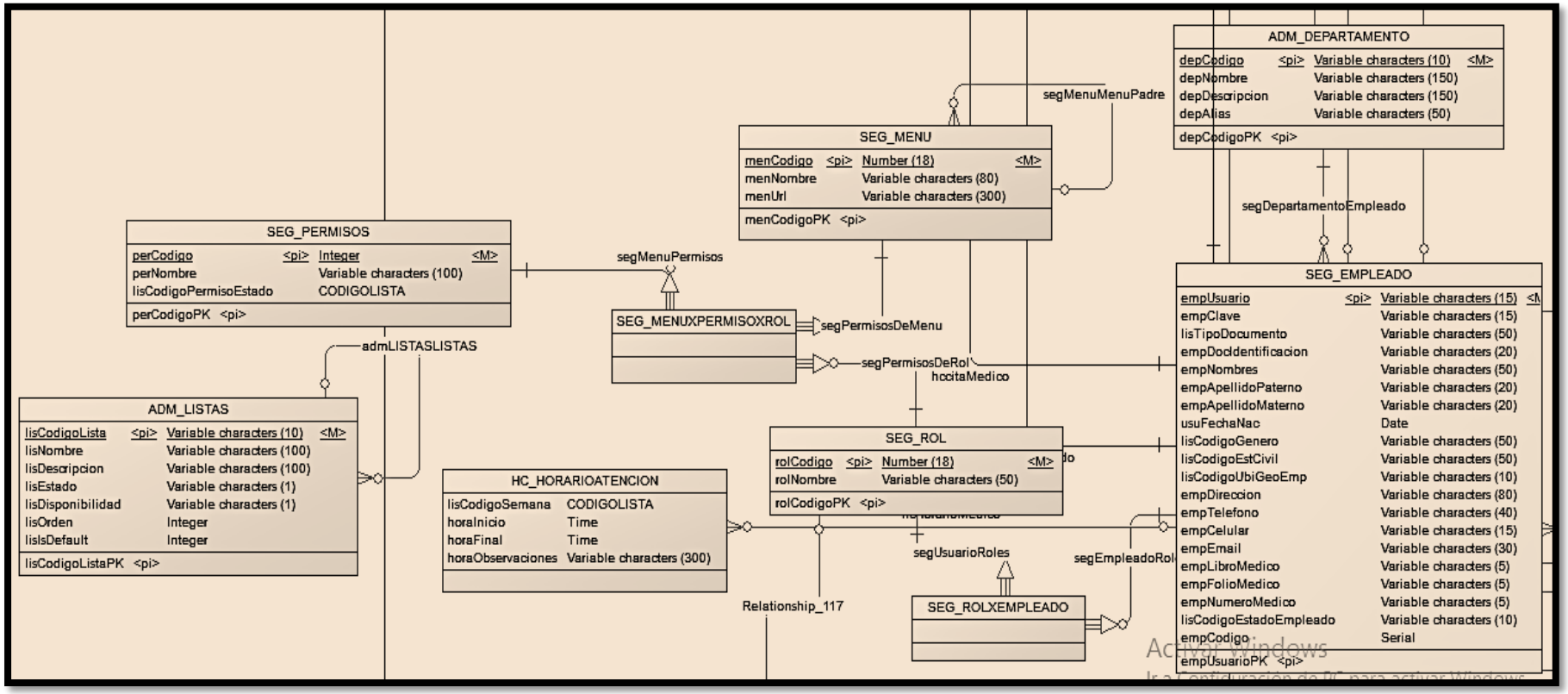


Gráfico 15 Modulo de Seguridades
Fuente: Autores

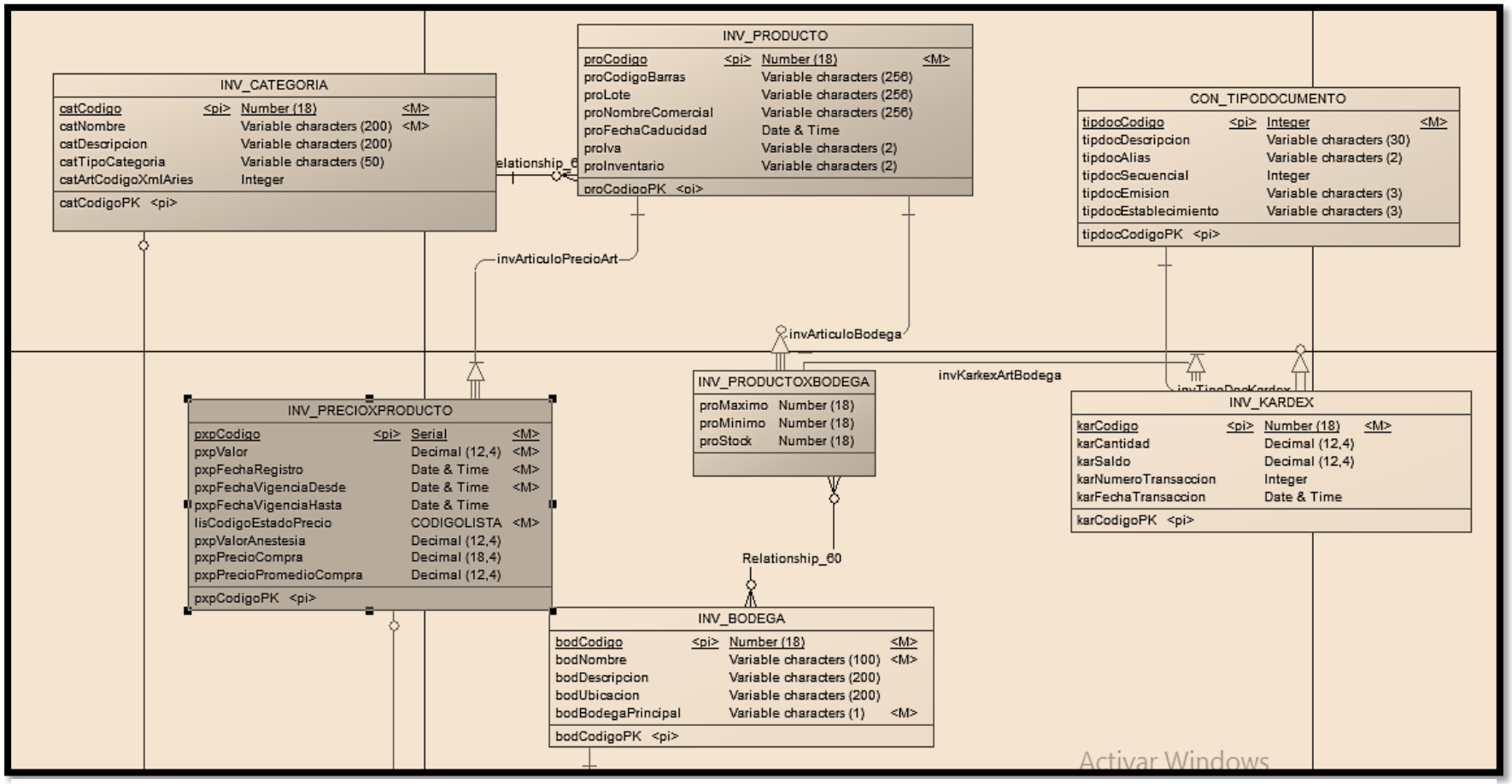


Gráfico 16 Modulo Gestión de Producto
Fuente: Autores

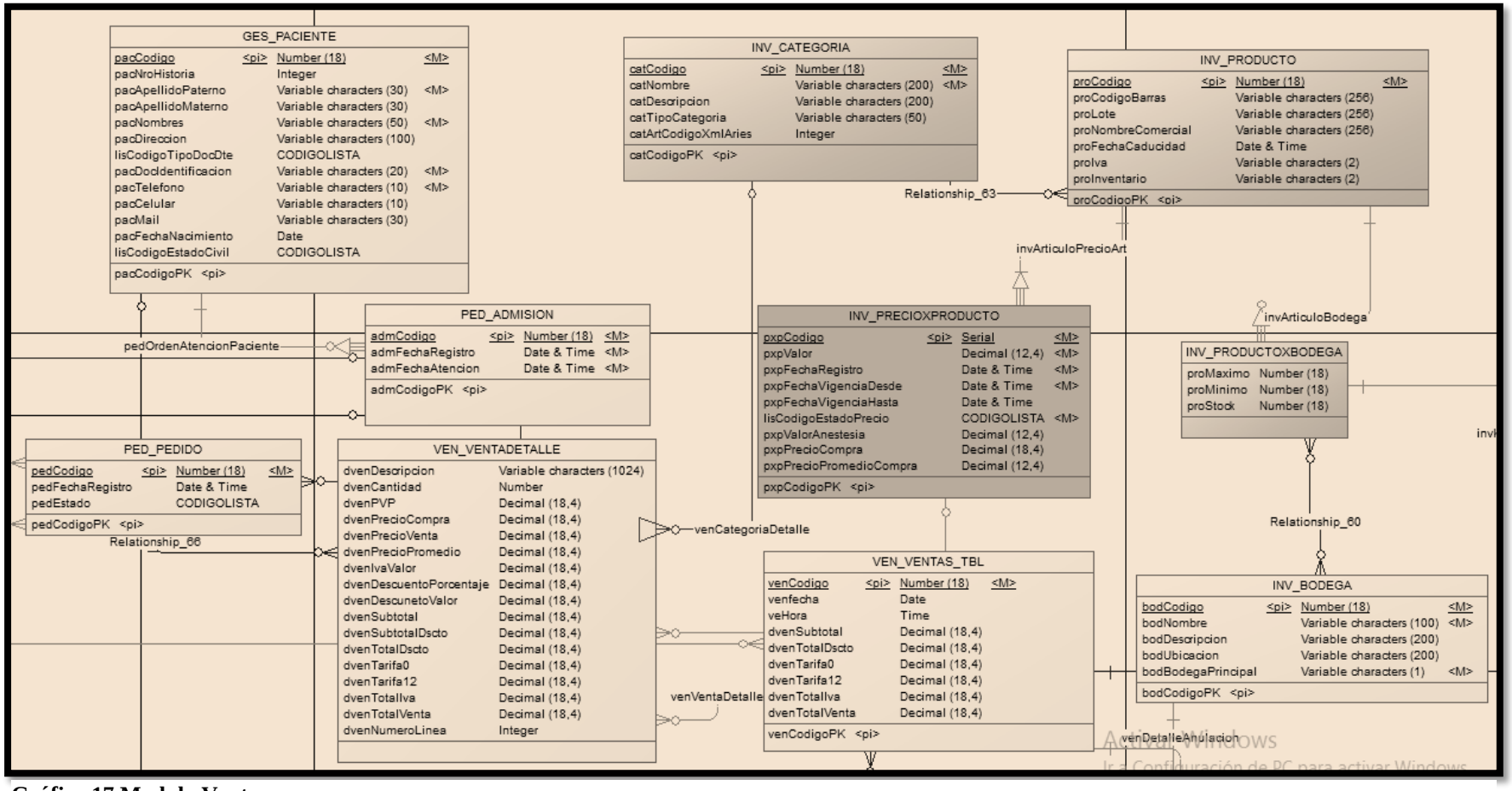


Gráfico 17 Modulo Ventas
Fuente: Autores

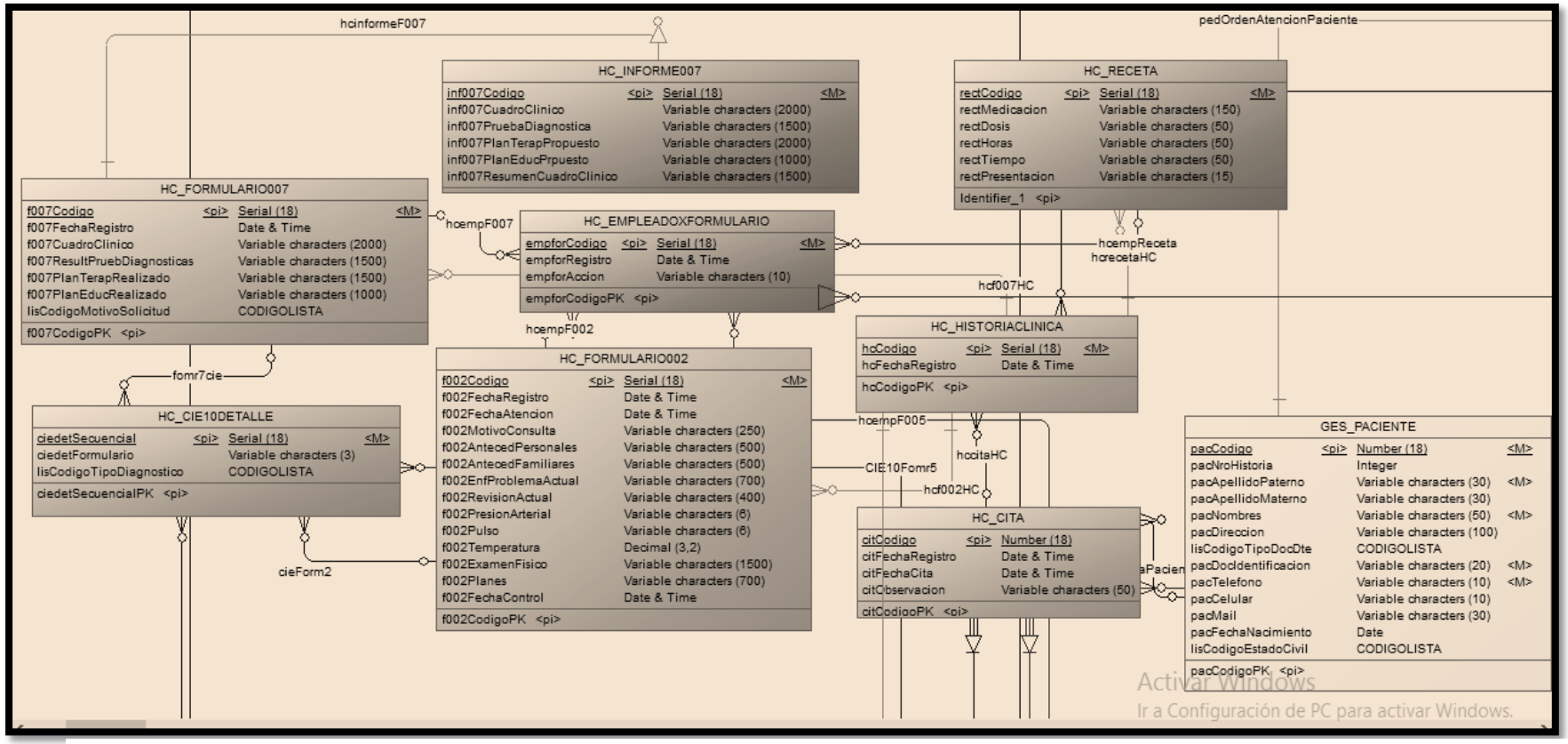


Gráfico 18 Modulo Historia Clínica
Fuente: Autores

4.3 DICCIONARIO DE DATOS

EMPLEADO		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
EMPUSUARIO	varchar(15)	PK (Primary Key)
ESPCODIGOESP	varchar(5)	
DEPCODIGO	varchar(10)	
EMPCLAVE	varchar(15)	
LISTIPODOCUMENTO	varchar(50)	
EMPDOCIDENTIFICACION	varchar(20)	
EMPNOMBRES	varchar(50)	
EMPAPELLIDOPATERNO	varchar(20)	
EMPAPELLIDOMATERNO	varchar(20)	
USUFECHANAC	datetime	
LISCODIGOGENERO	varchar(50)	
LISCODIGOESTCIVIL	varchar(50)	
EMPDIRECCION	varchar(80)	
EMPTELEFONO	varchar(40)	
EMPCELULAR	varchar(15)	
EMPEMAIL	varchar(30)	
EMPLIBROMEDICO	varchar(5)	
EMPFOLIOMEDICO	varchar(5)	
EMPNUMEROMEDICO	varchar(5)	
LISCODIGOESTADOEMPLEADO	varchar(10)	

Tabla 20 Empleado
Fuente: Autores

DEPARTAMENTO		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
DEPCODIGO	varchar(10)	PK (Primary Key)
DEPNOMBRE	varchar(150)	
DEPDESCRIPCION	varchar(150)	
DEPALIAS	varchar(50)	

Tabla 21 Departamento
Fuente: Autores

ESPECIALIDAD		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
ESPCODIGOESP	varchar(5)	PK (Primary Key)
ESPNOMBREESP	varchar(30)	

ESPDESCRIPCION	varchar(150)	
LISCODIGOTIPOESP	varchar(10)	‘

Tabla 22 Especialidad
Fuente: Autores

MENU		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
MENCODIGO	numeric(18, 0)	PK (Primary Key)
MENCODIGOPADRE	numeric(18, 0)	FK (Foring Key)
MENNOMBRE	varchar(80)	
MENURL	varchar(300)	

Tabla 23 Menú
Fuente: Autores

ROL		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
ROLCODIGO	numeric(18, 0)	PK (Primary Key)
ROLNOMBRE	varchar(50)	

Tabla 24 Rol
Fuente: Autores

ROLXEMPLEADO		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
ROLCODIGO	numeric(18, 0)	PK FK
EMPUSUARIO	varchar(15)	PK FK

Tabla 25 Rol por Usuario
Fuente: Autores

MENUXPERMISOXROL		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
ROLCODIGO	numeric(18, 0)	PK FK
MENCODIGO	numeric(18, 0)	PK FK
PERCODIGO	numeric(18, 0)	PK FK

Tabla 26 Menú por Permiso por Rol
Fuente: Autores

PACIENTE		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
PACCODIGO	numeric(18, 0)	PK (Primary Key)
PACNROHISTORIA	numeric(18, 0)	
PACAPELLIDOPATERNO	varchar(30)	
PACAPELLIDOMATERNO	varchar(30)	
PACNOMBRES	varchar(50)	

PACDIRECCION	varchar(100)	
LISCODIGOTIPODOCDTE	CODIGOLISTA:varchar(10)	
PACDOCIDENTIFICACION	varchar(20)	
PACTELEFONO	varchar(10)	
PACCELULAR	varchar(10)	
PACMAIL	varchar(30)	
PACFECHANACIMIENTO	Datetime	
LISCODIGOESTADOCIVIL	CODIGOLISTA:varchar(10)	
LISCODIGOGENERO	CODIGOLISTA:varchar(10)	

Tabla 27 Paciente
Fuente: Autores

PROVEEDOR		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
PROVCODIGO	varchar(20)	PK(Primary Key)
PROVRAZONSOCIAL	varchar(150)	
LISCODIGOTIPOIDEN	CODIGOLISTA:varchar(10)	
PROVDIRECCION	varchar(150)	
PROVTELEFONO	varchar(150)	

Tabla 28 Proveedor
Fuente: Autores

CLIENTE		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
CLICODIGO	varchar(20)	PK (Primary Key)
CLINOMBRE	varchar(150)	
LISCODIGOTIPOIDEN	CODIGOLISTA:varchar(10)	
CLIDIRECCION	varchar(100)	
CLITELEFONO	varchar(10)	

Tabla 29 Datos Cliente
Fuente: Autores

ADMISION		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
PACCODIGO	numeric(18, 0)	PK FK
ADMCODIGO	numeric(18, 0)	PK(Primary Key)
ADMFECHAREGISTRO	Datetime	
ADMFECHAATENCION	Datetime	

Tabla 30 Admisión
Fuente: Autores

PEDIDO		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
PEDCODIGO	numeric(18, 0)	PK (Primary Key)
SERCODIGO	numeric(18, 0)	FK (Foring Key)
PACCODIGO	numeric(18, 0)	FK (Foring Key)
ADMCODIGO	numeric(18, 0)	FK (Foring Key)
EMPUSUARIO	varchar(15)	FK (Foring Key)
PEDFECHAREGISTRO	datetime	
PEDESTADO	CODIGOLISTA:varchar(10)	

Tabla 31 Pedido
Fuente: Autores

VENTA		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
VENCODIGO	numeric(18, 0)	PK (Primary Key)
EMPUSUARIO	varchar(15)	FK (Foring Key)
CLICODIGO	varchar(20)	FK (Foring Key)
VENFECHA	datetime	
VEHORA	datetime	
DVENSUBTOTAL	decimal(18, 4)	
DVENTOTALDSCTO	decimal(18, 4)	
DVENTARIFA0	decimal(18, 4)	
DVENTARIFA12	decimal(18, 4)	
DVENTOTALIVA	decimal(18, 4)	
DVENTOTALVENTA	decimal(18, 4)	

Tabla 32 Venta
Fuente: Autores

VENTADETALLE		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
PEDCODIGO	numeric(18, 0)	
CATCODIGO	numeric(18, 0)	
PROCODIGO	numeric(18, 0)	
PXPCODIGO	numeric(18, 0)	
VENCODIGO	numeric(18, 0)	
DVENDESCRIPCION	varchar(1024)	
DVENCANTIDAD	numeric(18, 0)	
DVENPVP	decimal(18, 4)	
DVENPRECIOCOMPRA	decimal(18, 4)	
DVENPRECIOVENTA	decimal(18, 4)	
DVENPRECIOPROMEDIO	decimal(18, 4)	
DVENIVAVALOR	decimal(18, 4)	

DVENDESCUENTOPORCENTAJE	decimal(18, 4)	
DVENDESCUNETOVALOR	decimal(18, 4)	
DVENSUBTOTAL	decimal(18, 4)	
DVENSUBTOTALDSCTO	decimal(18, 4)	
DVENTOTALDSCTO	decimal(18, 4)	
DVENTARIFA0	decimal(18, 4)	
DVENTARIFA12	decimal(18, 4)	
DVENTOTALIVA	decimal(18, 4)	
DVENTOTALVENTA	decimal(18, 4)	

Tabla 33 Detalle de Venta
Fuente: Autores

CITA		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
EMPUSUARIO	varchar(15)	PK FK
SEG_EMPUSUARIO	varchar(15)	PK FK
CITCODIGO	numeric(18, 0)	PK
PACCODIGO	numeric(18, 0)	PK
ADMCODIGO	numeric(18, 0)	PK
CITFECHAREGISTRO	Datetime	
CITFECHACITA	Datetime	
CITOBSERVACION	varchar(50)	

Tabla 34 Cita
Fuente: Autores

HORARIO DE ATENCIÓN		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
EMPUSUARIO	varchar(15)	PK(Primary Key)
LISCODIGOSEMANA	CODIGOLISTA:varchar(10)	
HORAINICIO	Datetime	
HORAFINAL	Datetime	
HORAOBSERVACIONES	varchar(300)	

Tabla 35 Horario Atención
Fuente: Autores

FORMULARIO002		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
F002CODIGO	numeric(18, 0)	PK (Primary Key)
HCCODIGO	numeric(18, 0)	FK (Foring Key)
F002FECHAREGISTRO	datetime	
F002FECHAATENCION	datetime	
F002MOTIVOCONSULTA	varchar(250)	

F002ANTECEDPERSONALES	varchar(500)	
F002ANTECEDFAMILIARES	varchar(500)	
F002ENFPROBLEMAACTUAL	varchar(700)	
F002REVISIONACTUAL	varchar(400)	
F002PRESIONARTERIAL	varchar(6)	
F002PULSO	varchar(6)	
F002TEMPERATURA	decimal(3, 2)	
F002EXAMENFISICO	varchar(1500)	
F002PLANES	varchar(700)	
F002FECHACONTROL	datetime	

Tabla 36 Formulario002
Fuente: Autores

FORMULARIO005		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
F005CODIGO	numeric(18, 0)	PK (Primary Key)
HCCODIGO	numeric(18, 0)	FK (Foring Key)
F005FECHAREGISTRO	Datetime	
F005FECHAINICATENCION	Datetime	
F005FECHAFINATENCION	Datetime	
F005EVOLUCION	varchar(3500)	
F005PRESCRIPCIONES	varchar(1000)	

Tabla 37 Formulario005
Fuente: Autores

FORMULARIO005		
CAMPO	**TIPO DATO**	**TIPO CAMPO**
F007CODIGO	numeric(18, 0)	PK
HC__F007CODIGO	numeric(18, 0)	FK
INF007CODIGO	numeric(18, 0)	
HCCODIGO	numeric(18, 0)	
F007FECHAREGISTRO	datetime	
F007CUADROCLINICO	varchar(2000)	
F007RESULTPRUEBDIAGNOSTICAS	varchar(1500)	
F007PLANTERAPREALIZADO	varchar(1500)	
F007PLANEDUCREALIZADO	varchar(1000)	
LISCODIGOMOTIVOSOLICITUD	CODIGOLISTAvarchar (10)	

Tabla 38 Formulario007
Fuente: Autores

4.4 Diagrama De La Arquitectura Del Sistema

Para la elaboración del proyecto se aplicó la programación por capas, el objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos.

(https://es.slideshare.net/Decimo/arquitectura-3-capas)

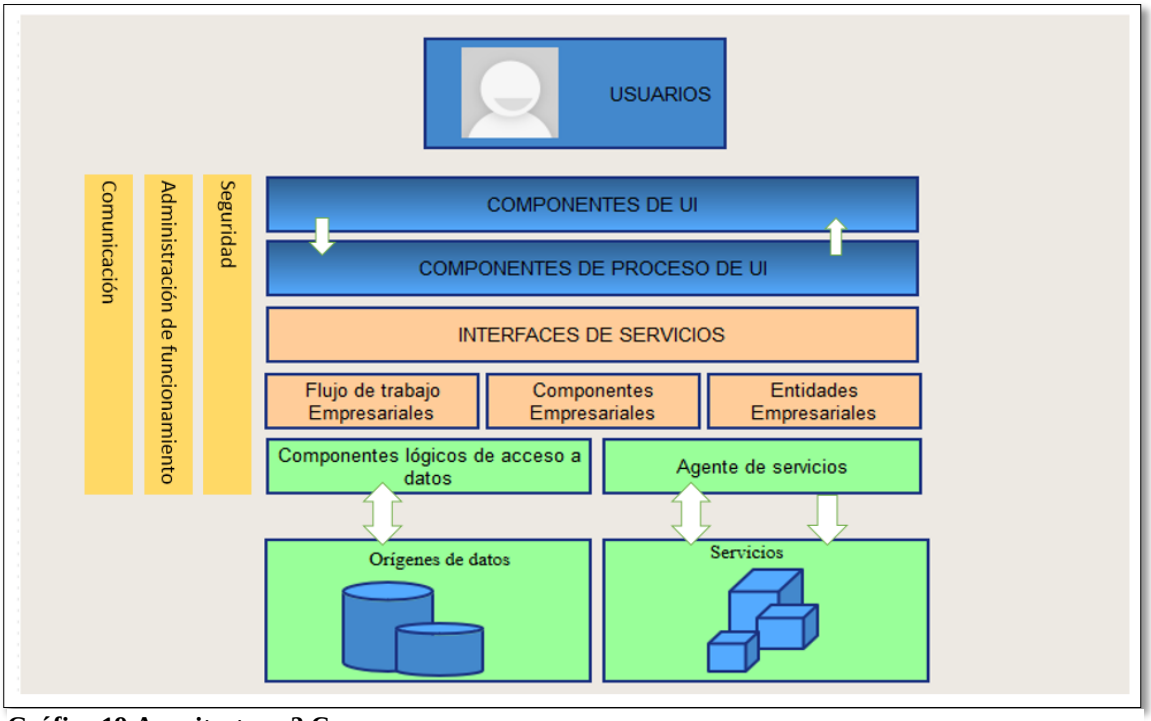


Gráfico 19 Arquitectura 3 Capas
Fuente: Autores

El sistema se desarrolla en arquitectura 3 capas en un entorno de desarrollo en Windows XP, en el lenguaje de C Sharp .net y el gestor de base Microsoft SQL Server 2008 y UI DevExpress.

4.4.1 Capa de Presentación.

La capa de presentación o interfaz de usuario es el mecanismo de interacción del usuario con el sistema. Aquí se definirán las interfaces mediante las cuales el usuario accederá al sistema para gestionar la información. Para esta capa es necesario el diseño de prototipos que serán aprobados por el usuario final.

En el gráfico 21 (Arquitectura 3 capas), se puede apreciar los Componentes de UI (Usuario Interface) y Componentes de Proceso de UI. Estos forman parte de la capa de presentación del proyecto.

Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" para el usuario, generalmente se presentan como formularios.

4.4.2 Capa de Negocio.

En la Capa de Negocio, se definirán todos los algoritmos y funcionalidades que son parte del giro del negocio.

4.4.3 Capa de Datos

Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de base de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Esta capa estará encargada de gestionar toda la DATA que manejará el sistema, la DATA se almacenará en SQL SERVER 2008 y se utilizarán las funcionalidades propias del motor como:

- Procedimientos
- Disparadores

- Índices
- Jobs automatizados
- Backups automáticos
- Tablas particionadas

4.5 Diseño De Interfaces

A continuación se presenta el diseño de interfaces del sistema.

PANTALLA INICIAL

Diseño de Inicio de sesión

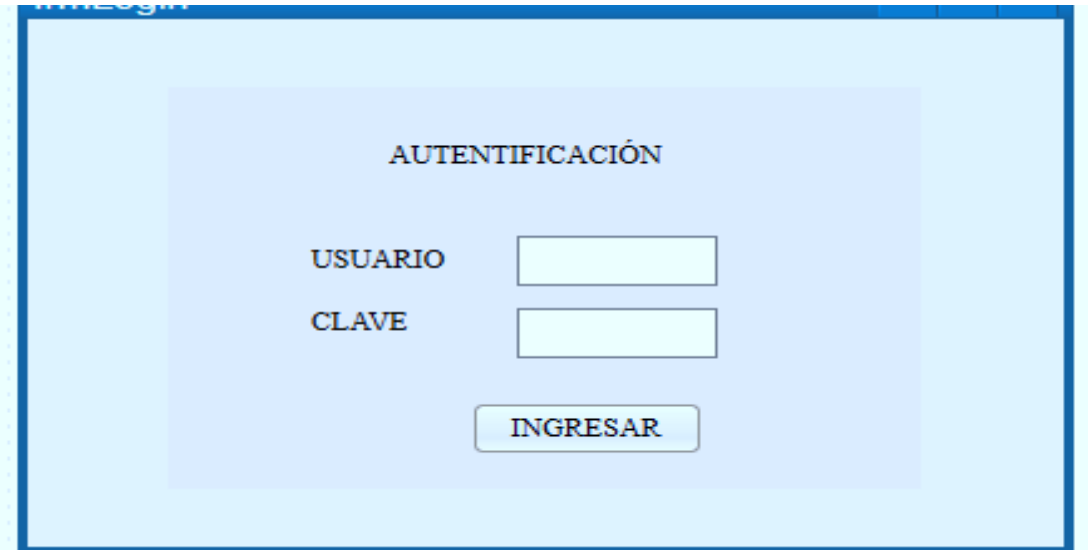


Gráfico 20 Interfaz – Inicio de Sesión
Fuente: Autores

La interfaz está diseñada para que el empleado que pertenece al Consultorio Salud & Vida pueda fácilmente asociar el ingreso al sistema de la misma forma como lo realizaría en cualquier otro sistema.

SEGURIDADES

Empleado

Diseño de creación de empleados

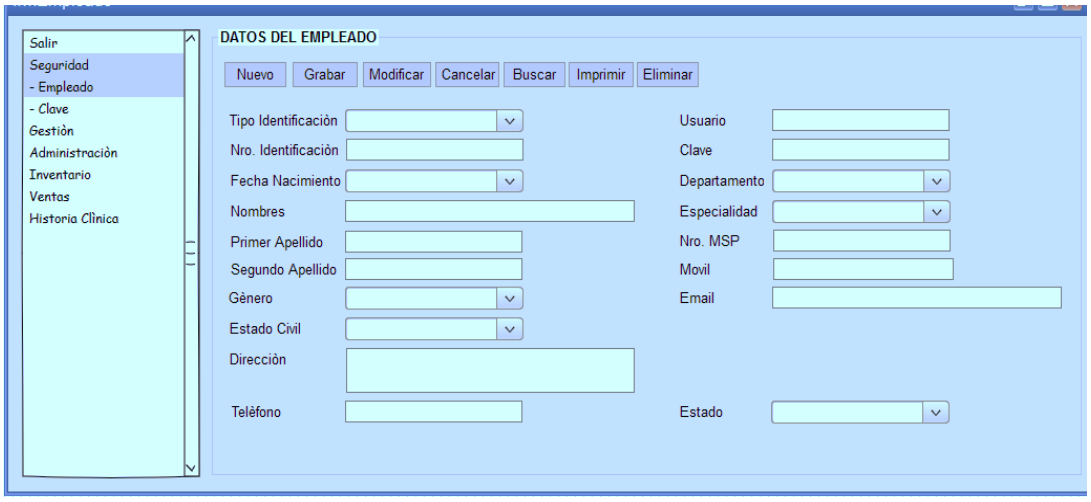


Gráfico 21 Interfaz – Empleado
Fuente: Autores

La interfaz está diseñada para que el administrador pueda crear al empleado y asignarle su usuario y clave.

Clave

Diseño de creación de cambio de clave

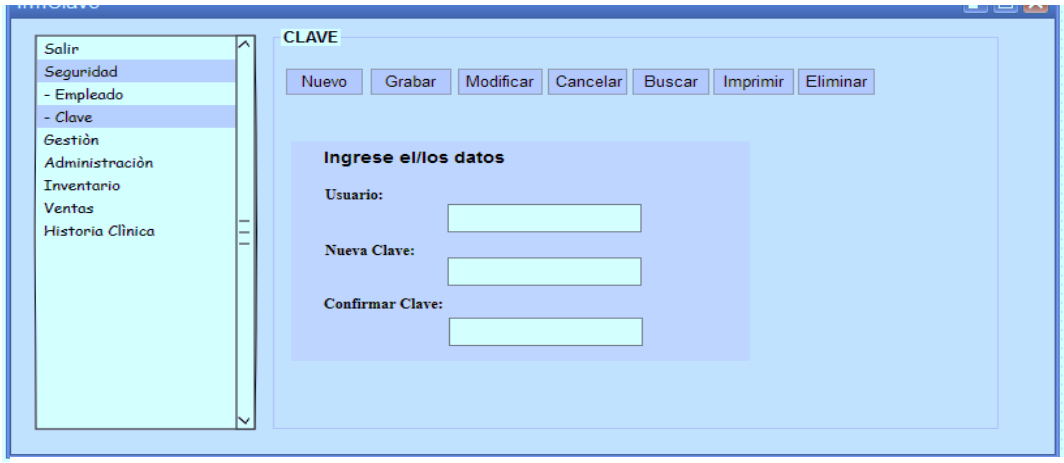


Gráfico 22 Interfaz – Clave
Fuente: Autores

La interfaz está diseñada para que el empleado pueda crear una nueva clave para su mayor seguridad.

GESTIÓN

Paciente

Diseño de creación de pacientes

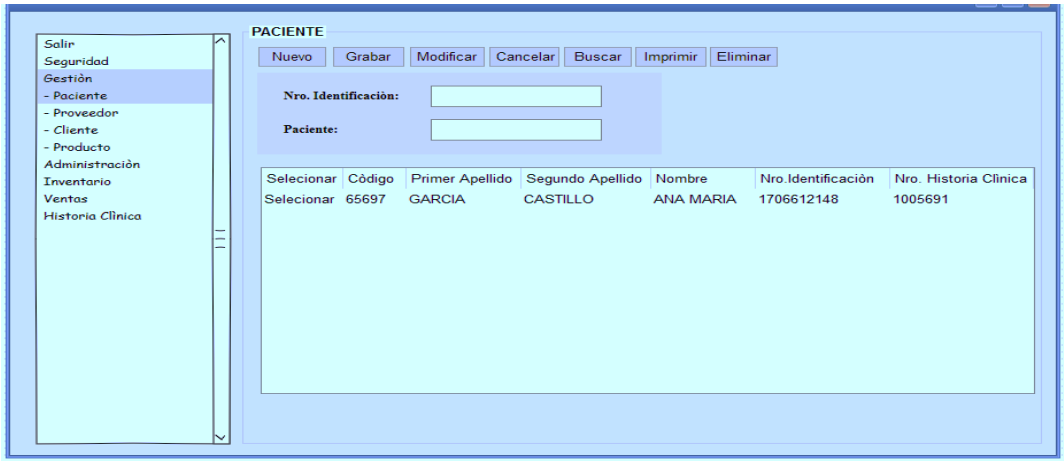


Gráfico 23 Interfaz – Paciente
Fuente: Autores

La interfaz está diseñada para que el usuario pueda crear pacientes que serán de utilidad para (Facturación, Cita Médica, Historia Clínica).

Proveedor

Diseño de creación de proveedores

Gráfico 24 Interfaz – Proveedor
Fuente: Autores

La interfaz está diseñada para que el usuario pueda crear proveedores que serán de utilidad para el ingreso de productos.

Cliente

Diseño de creación de clientes

Gráfico 25 Interfaz – Cliente
Fuente: Autores

La interfaz está diseñada para que el usuario pueda crear clientes que serán de utilidad para la facturación.

Producto

Diseño de creación de empleados

Gráfico 26 Interfaz – Producto
Fuente: Autores

La interfaz está diseñada para que el usuario pueda crear producto y servicios que serán de utilidad para la (Facturación, Ingreso/Egreso).

ADMINISTRACIÓN

Departamentos

Diseño de creación de departamentos

Gráfico 27 Interfaz – Departamento
Fuente: Autores

La interfaz está diseñada para que el usuario pueda crear los departamentos que serán de utilidad para asignarle el departamento al que pertenece el empleado.

Especialidades

Diseño de creación de especialidades

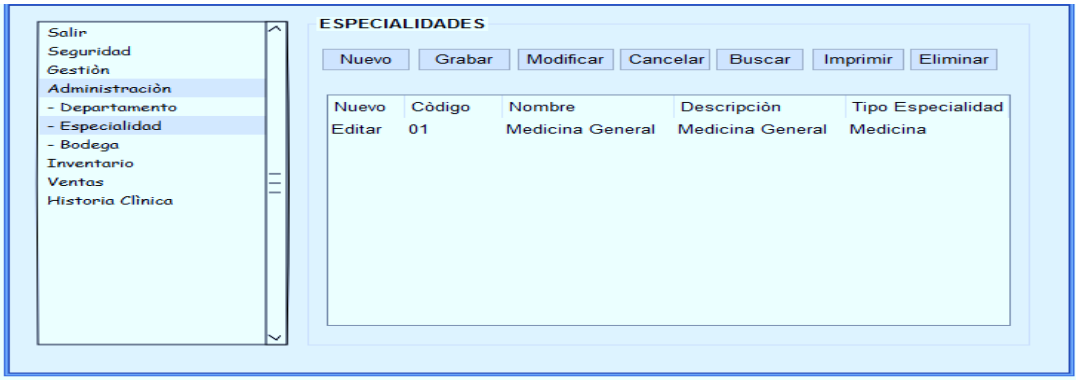


Gráfico 28 Interfaz – Especialidad
Fuente: Autores

La interfaz está diseñada para que el usuario pueda crear las especialidades que serán de utilidad para asignarle al Médico.

Bodega

Diseño de creación de bodegas

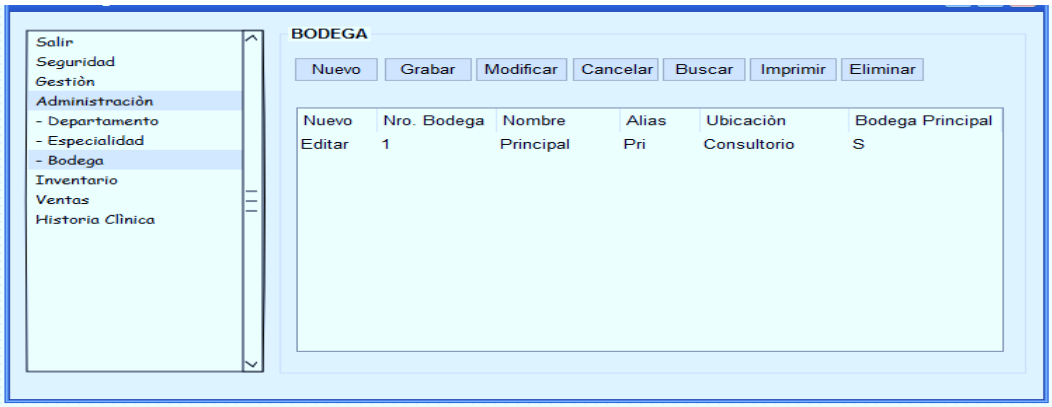


Gráfico 29 Interfaz – Bodega
Fuente: Autores

La interfaz está diseñada para que el usuario pueda crear la bodega que será de utilidad para asignarle al Producto.

INVENTARIO

Ingreso / Egreso

Diseño de creación de ingreso-egreso de productos

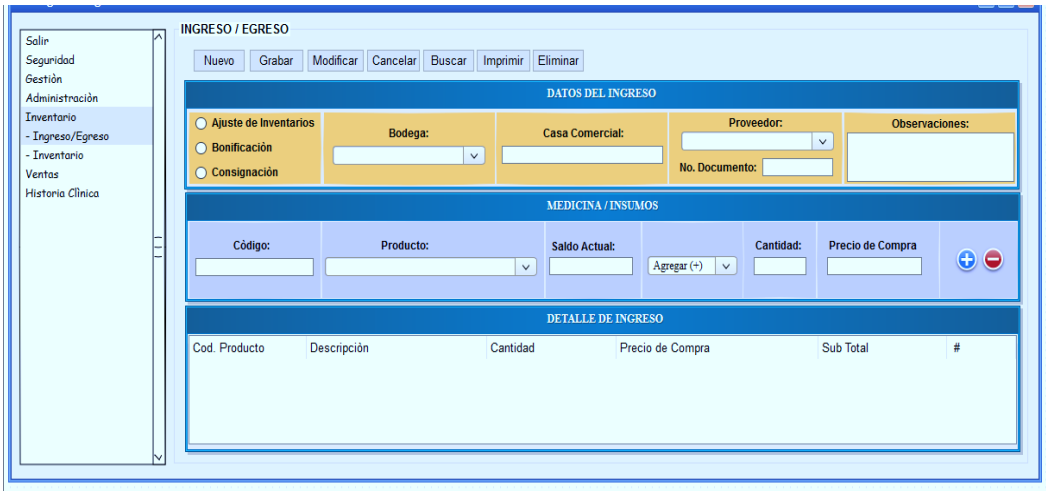


Gráfico 30 Interfaz – Ingreso/Egreso Inventario
Fuente: Autores

La interfaz está diseñada para que el usuario pueda ingresar las medicinas e insumos que compre el administrador para el tratamiento de los pacientes.

VENTAS

Facturación

Diseño de creación de facturación

Gráfico 31 Interfaz – Facturación
Fuente: Autores

La interfaz está diseñada para que el usuario pueda realizar la emisión de facturas por el servicio de salud brindado.

Anulación De Ventas

Diseño de creación de anulación de ventas

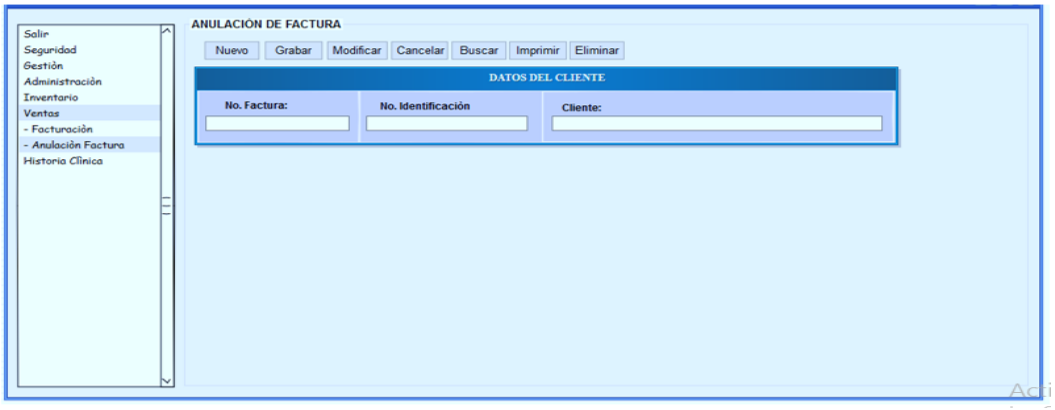


Gráfico 32 Anulación de Facturación
Fuente: Autores

La interfaz está diseñada para que el usuario pueda realizar la anulación de facturas ya sea por cambio de datos y/o equivocación del usuario.

HISTORIA CLÍNICA

Horario De Atención

Diseño de creación de horario de atención del médico

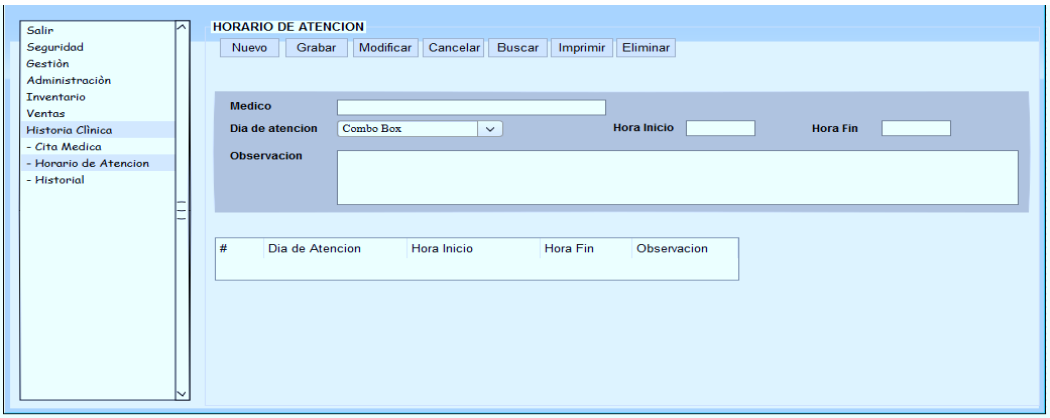


Gráfico 33 Horario de Atención del Médico
Fuente: Autores

La interfaz está diseñada para que el usuario pueda realizar la creación del horarios de atención y los días que el Médico dará la consulta.

Cita Médica

Diseño de creación de anulación de ventas

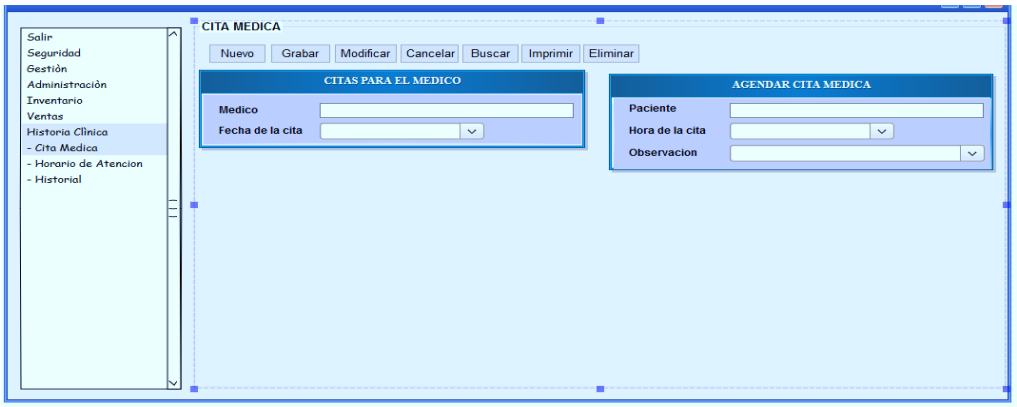


Gráfico 34 Interfaz – Cita Médica
Fuente: Autores

La interfaz está diseñada para que el usuario pueda realizar la creación de la cita médica y quede reservada hasta el día de su atención.

Historial Clínico

Diseño de creación de historial clínico del paciente

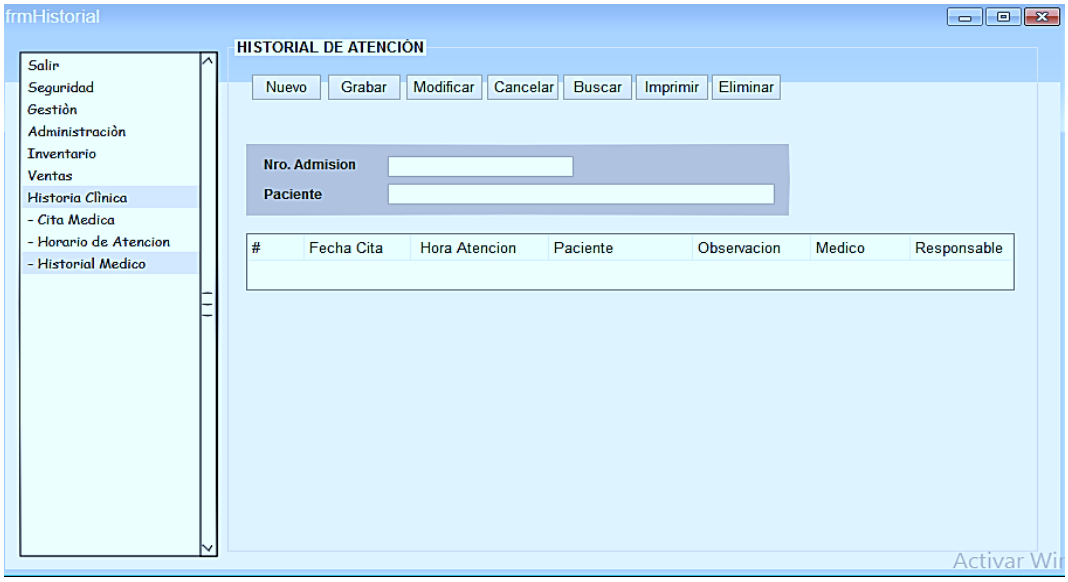


Gráfico 35 Historial de Atención del Paciente
Fuente: Autores

La interfaz está diseñada para que el Medico pueda crear la historia clínica del paciente donde la primera vez que acude el paciente al consultorio Salud & Vida debe llenar (Hoja de Consulta Externa HCU_Form002) y si el paciente es subsecuente debe llenar (Hoja de Nota de Evolución HCU_Form005), en los dos formularios el medico podrá ubicar los diagnósticos presuntivos y/o definitivos acorde a la CIE-10 (Código Internacional de Enfermedades).

4.6 Administración y Seguridad.

Para mayor seguridad el sistema contará con la autentificación de usuario y contraseña para el acceso al sistema.

Usuario. Al momento de ingresar al sistema se deberá ingresar el usuario correcto.

Contraseña.- Para acceder al sistema se debe ingresar una contraseña que el administrador le asignará previamente.

ROL	PERFILES	FUNCIONES
Administrador	Acceso a todos los módulos	Administra todo el sistema
Enfermería	Al módulo de facturación	Realiza facturación y ventas

Tabla 39 Funcionalidades
Fuente: Autores

4.7 Estándares de Programación Utilizados.

Se creará una sola solución con un solo proyecto que dentro del mismo estarán las carpetas identificando a los módulos, sub-módulos, carpetas y demás objetos.

Para nombrar cualquier objeto del proyecto se deben tomar en cuenta las siguientes consideraciones generales:

- Los nombres de los objetos deben ser descriptivos y fáciles de entender.
- El nombre de un objeto no debe contener palabras reservadas o caracteres especiales.
- No deben contener espacios en blanco, de ser necesaria la separación de palabras se debe utilizar un guion bajo.
- Los objetos deben nombrarse en idioma Español.

4.7.1 Prefijos

Todo objeto del proyecto debe utilizar un prefijo estándar que permita relacionarlo, mismo que corresponderá a las tres primeras letras de su nombre.

A continuación se indican los prefijos de cada uno de los objetos más utilizados:

OBJETOS	PREFIJO
FORMULARIOS	frm
CLASES	cls
REPORTES	rpt
ESTILOS	est
DATASET	dst
TextBox	txt
DropDownList	ddl
Gridview	grd
Label	Lbl
Boton	Btn
CheckBox	Chk
RadioButtom	Opt

Tabla 40 Prefijos - Programación
Fuente: Autores

4.7.2 Objetos del proyecto - Formularios

Para nombrar los formularios se deben tener en cuenta las siguientes consideraciones particulares:

- Al igual que para el resto de objetos, el nombre de un formulario debe ser expresado utilizando palabras completas y debe ser lo más descriptivo posible.
- Debe ser expresado en singular. Si el nombre del formulario contiene varias palabras, deben estar unidas y las dos o más palabras expresadas en singular y la primera letra en mayúsculas acorde con lo que sea correcto en el idioma español.

Ejemplos:

Formulario ingreso de pacientes:

frmPaciente

Reporte de ventas:

rptFacturaVenta

Los formularios deben contener comentarios descriptivos y escritos como tipo de oración.

4.7.3 Objetos de los Formularios

Para nombrar los objetos que serán utilizados en los formularios se deben tener en cuenta las siguientes consideraciones particulares:

- El nombre de un objeto debe ser único dentro del esquema en el que se encuentra
- Al igual que para el resto de objetos, el nombre debe ser expresado utilizando palabras completas y debe ser lo más descriptivo posible.
- Debe ser expresado en singular. Si el nombre del objeto contiene varias palabras, deben estar unidas y las dos o más palabras expresadas en singular y la primera letra en mayúsculas acorde con lo que sea correcto en el idioma español.

Ejemplos:

Campo/Objeto: Nombre

Caja de Texto

- txtNombre

Campo/Objeto: Nombre

Gridview:

- grdDetalleFactura

Los campos deben contener comentarios descriptivos y escritos como tipo de oración

4.7.4 Clases

A excepción de las clases que se autogeneran con LINQ y las clases que se generan con los formularios, las demás, es decir las creadas por el desarrollador deben guardar el siguiente formato:

- Iniciar el prefijo con las letras cls
- El nombre de una clase debe ser expresada utilizando palabras completas y debe ser lo más descriptivo posible.
- Debe ser expresado en singular. Si el nombre de la clase contiene varias palabras, deben estar unidas y las dos o más palabras expresadas en singular y la primera letra en mayúsculas acorde con lo que sea correcto en el idioma español.

Ejemplos:

Clase pacientes:

clsPaciente

Las clases deben contener comentarios descriptivos y escritos como tipo de oración.

4.7.5 Procedimientos

A excepción del procedimiento Page_Load y los generados automáticamente por LINQ, todos los demás procedimientos deberán tener el formato:

- Iniciar el prefijo con las letras prc
- Al igual que para el resto de objetos, el nombre de un procedimiento debe ser expresado utilizando palabras completas y debe ser lo más descriptivo posible.
- Debe ser expresado en singular. Si el nombre del procedimiento contiene varias palabras, deben estar unidas y las dos o más palabras expresadas en singular y la primera letra en mayúsculas acorde con lo que sea correcto en el idioma español.

Ejemplos:

Procedimiento para ingresar el paciente:

prcSegModificarUsuario ()

4.7.6 Funciones

A excepción de las funciones generadas automáticamente por LINQ, todas los demás funciones deberán tener el formato:

- El prefijo con las letras fnc
- El nombre de una función debe ser expresada utilizando palabras completas y debe ser lo más descriptivo posible.
- Debe ser expresado en singular. Si el nombre de la función contiene varias palabras, deben estar unidas y las dos o más palabras expresadas en singular y la primera letra en mayúsculas acorde con lo que sea correcto en el idioma español.

Ejemplos:

Función para validar que exista el paciente:

fncValidaExistaPaciente()

Solución				
Proyecto				
	Ventas			
		Común		
			Clase	
		Datos		
			Clase	
		Negocio		
			Clase	
		Presentación		
			Administración	
				frmPaciente
				frmCliente
			Venta	
				frmVenta
				frmAnulacionFactura
			Imágenes	
				imgNuevo.jpg
				imgGrabar.jpg
				imgImprimir.jpg

Tabla 41 Árbol estructural del Proyecto/Solución
Fuente: Autores

4.7.7 Declaración de variables

Para nombrar a las variables que se usarán en los formularios se deben tener en cuenta las siguientes consideraciones particulares:

- El nombre de una variable debe ser único dentro del esquema en el que se encuentra
- Al igual que para el resto de variables el nombre debe ser expresado utilizando palabras completas y debe ser lo más descriptivo posible.
- Debe ser expresado en singular. Si el nombre de la variable contiene varias palabras, deben estar unidas y las dos o más palabras expresadas en singular y la primera letra en mayúsculas acorde con lo que sea correcto en el idioma español.

VARIABLES	PREFIJO
Integer	Int
Long	lng
Bolean	bln
Object	obj
String	Str
Double	Dbl

Tabla 42 Prefijo Variables
Fuente: Autores

Ejemplos:

Variable: Auxiliar

Variable de tipo texto/string

- strAuxiliar

Variable: Auxiliar

Variable de tipo integer/int

- intAuxiliar

Campo/Objeto: Auxiliar

Variable de tipo Double/Decimal

- dblAuxiliar

Comentarios de ayuda

Al comenzar a desarrollar un formulario, clase o cualquier otra definición se debe comentar indicando para que sirve o en que se va a utilizar dicho código fuente.

Indicar lo siguiente: Desarrollado por, fecha de creación, descripción

Ejemplo:

//Descripción: Describir la función del código ejemplo (//Define y retorna la estructura de la tabla)

4.8 Pruebas Realizadas

Se realizaron las pruebas de funcionalidad y trazabilidad del software en conjunto con la Dra. Katya Caisa, la misma que probo el sistema con su usuario dando como resultado la aprobación del mismo, por tanto se determinó que el sistema cumple con lo solicitado por la Doctora para el correcto funcionamiento del Consultorio Salud & Vida.

Funcionales

RF01: El usuario podrá acceder a la opción de citas médicas del rol que se creara para dicho fin.

Detalle RF01: Se realiza la creación del ROL para la opción de cita médica. El Rol creado es asignado al usuario. Una vez realizas estas actividades el usuario procede a realizar el inicio de sesión. Accede al sistema y mostrara el menú acorde a los roles asignados.

RF02: El usuario podrá realizar consulta de ventas y posteriormente exportar a Excel.

Detalle RF02: El usuario escoge la opción de consulta de ventas la cual puede ser de 3 modos (por fechas, por fechas y paciente o cliente, por fechas y número factura)

RF03: El usuario podrá modificar la información del producto y/o servicio del rol que se creara para dicho fin.

Detalle RF03: Se realiza la creación del ROL para la opción de producto. El usuario podrá modificar datos esenciales como el precio de venta acorde a la variación del mercado.

RF04: El usuario podrá consultar los honorarios médicos

Detalle RF04: El usuario administrador escoge la opción de consulta de honorarios médicos el cual permite realizar el pago a los médicos que prestaron sus servicios profesionales dentro del Consultorio Salud & Vida dando clic en el botón de dólares.

No Funcionales

RNF01: Usabilidad.- Las interfaces creadas mantienen el diseño estándar de uso familiar para el usuario.

Detalle RNF01: Se revisa con el usuario las características estándares que provee el Sistema, filtros de consulta, (fuente de datos) que contiene la interfaz, ocultar controles.

RNF02: Plataforma.- El software se encuentra instalado en plataformas Windows.

Detalle RNF02: Se revisa con los usuarios la plataforma bajo la cual funciona el sistema.

RNF03: Rendimiento.- La arquitectura que se encuentra establecida está cuantificada para grandes manejos de información.

Detalle RNF03: Se revisa con los usuarios claves la transaccionalidad de información teniendo un resultado de tiempos satisfactorios.

RNF04: Desempeño.- Las opciones creadas no presentaran problemas para su manejo e implementación.

Detalle RNF04: Se revisa con los usuarios claves la funcionalidad de las opciones no presentan anomalías.

4.8.1 Caja Blanca

Esta prueba realizada al sistema de gestión de historial clínico y facturación, está basada a todas las pruebas de código y procesos internos del sistema, tales como los procesos de historia clínica y control de facturación automatizada y funciones que se encuentren ejecutándose correctamente.

4.8.2 Caja negra.

Esta prueba realizada al sistema de gestión de historial clínico y facturación, está basada a todos los procesos externos del sistema, tales como los ingresos de datos informativos de cada paciente.

4.9 Implementación

De acuerdo a las pruebas realizadas al sistema de gestión de historial clínico y facturación para el Consultorio Salud & Vida, los requerimientos y equipos que debe tener el consultorio de salud son los siguientes:

4.9.1 Requerimientos De HW/SW

Hardware

1. Procesador CORE I5, 3 GHz.
2. Disco duro de 1TB.
3. Memoria DDR4 4GB.

Software

1. Sistema operativo Microsoft Windows XP, Service Pack 2.

2. Microsoft Visual Studio 2010
3. Microsoft SQL Server 2008
4. DevExpress Versión 14.2.6

4.9.2 Instalación y Configuración del Sistema

Pasos para la instalación y configuración del sistema propuesto:

- Instalar el Gestor de Base de Datos SQL Server 2008
- Instalar el ejecutable del programa realizado en C#.net
- Instalar el DevExpress Versión 14.2.6

4.10 Mantenimiento

Las políticas de mantenimiento tanto correctivas como preventivas para el software.

Mantenimiento Correctivo

El programador tiene el deber de ayudar al usuario si ocurre un problema en el sistema.

Mantenimiento Preventivo

Mantenimiento de las tablas bases o cuando tenga el usuario un error.

Manual de Usuario

El detalle de este punto se encuentra en el Anexo Manual de Usuario.

Manual Técnico

El detalle de este punto se encuentra en el Anexo Manual Técnico.

4.11 Plan de Capacitación

El plan de capacitación es socializar a los usuarios la utilización del sistema, los mismos que reciben la capacitación y luego ellos se encargan de capacitar a sus nuevos compañeros. En función a lo mencionado se puede apreciar en el manual de usuario las actividades de capacitación.

5 CONCLUSIONES

- Para el análisis de los inconvenientes y necesidades del consultorio clínico se realizó una entrevista, a fin de establecer la verdadera problemática, llegando así a concluir con la meta planteada y solventar los inconvenientes, mediante el desarrollo del sistema de gestión de historial clínico y facturación automatizada.

- Se estudió y se redefinió los procesos de registro y almacenamiento de la información, resultado del agendamiento de citas y de las historias clínicas de los pacientes, con la finalidad de obtener un sistema que le permita administrar adecuadamente dicha información y disminuir los tiempos y mejorar el servicio de atención de pacientes.

- Se desarrolló un sistema informático para el adecuado control de citas, historia médica y proceso de facturación, el sistema está desarrollado con arquitectura de 3 capas, herramientas de última generación C#.Net, Gestor de Base de Datos SQL Server 2008 y UI DevExpress, para mejorar los procesos del consultorio médico.

- Se efectuaron varias pruebas de los diferentes procesos que accede el sistema de Historial Clínico y Facturación, a fin de verificar posibles errores o mejoras a realizar, para así, lograr la adecuada funcionalidad de los distintos módulos que este contiene al servicio y uso de los usuarios del consultorio médico.

- Se implementó el sistema de Historial Clínico y Facturación en el consultorio médico Salud y Vida y se realizó la capacitación para el correcto manejo y uso del sistema, acorde al perfil de cada usuario.

6 RECOMENDACIONES.

- Después de haber concluido con el proyecto y desarrollo del sistema de gestión de historias clínicas y facturación automatizada se ha establecido las siguientes recomendaciones para una buena utilización del sistema.

- Se considera necesario para un proceso más rápido y efectivo, el uso de un equipo tecnológico con las siguientes características: procesador CORE I5, disco duro de 1TB y Memoria DDR4 4GB.

- Contratar Hosting que permita el agendamiento de citas por internet.

- Se recomienda una unidad de disco duro externo para el almacenamiento y respaldo de información.

7 Bibliografía

8. RAMIR, S. (2015). *Bases de Datos SGBD* (3 Edición, Italia ed.). Italia.

ARIAS, D. y. (1999). *"Métodos y Técnicas para la Investigación de las Ciencias Sociales"* (2da Edición ed.). México.

CLUM, N. (2006). *"Programación"* (Hamburgo 2da Edición ed.). Alemania.

Concepto de Programación. (08 de 2018). Obtenido de Equipo de Redacción de Concepto: https://concepto.de/programacion/

Debitoor. (s.f.). Obtenido de Factura: https://debitoor.es/glosario/definicion-factura

Definición de Servidor IIS. (s.f.). Obtenido de https://2003server.webcindario.com/iis/definici.htm

Duran, J. (2007). *El Lenguaje de Programacion C#*. Obtenido de Lenguaje C#: https://lenguajedeprogramacion.com/csharp/

García, K. O. (s.f.). *DevExpress*. Obtenido de el mejor aliado en el desarrollo: http://dawconsblog.blogspot.com/2014/04/devexpress-el-mejor-aliado-en-el.html

Harvey, D. (2007). *Programación orientada a objetos (POO)* (Segunda edición ed.).

Hill, G. M. (2014). *Base de Datos* (Tercera edición ed.). Editorial Concepción.

https://es.slideshare.net/Decimo/arquitectura-3-capas. (s.f.).

KENDALL, K. &. (s.f.). *Análisis y diseño de sistemas* (Tercera Edición ed.).

MICROSOFT. (2007). *Información general sobre ASP.NET*. Obtenido de Visual Studio 2010: https://msdn.microsoft.com/es-es/library/4w3ex9c2(v=vs.100).aspx

Microsoft SQL Server. (s.f.). Obtenido de https://www.ecured.cu/Microsoft_SQL_Server

Navas, P. (s.f.). *El ciclo de la vida en Internet*. Obtenido de El ciclo de vida: http://www.spw.cl/proyectos/apuntes2/cap_6.htm

Pública, M. d. (Agosto 2007). *Expediente Único para la* (4ª Edición ed.). Ecuador.

RIPOLL, L. Q. (s.f.). *SISTEMAS DE GESTIÓN DE*. Obtenido de http://api.eoi.es/api_v1_dev.php/fedora/asset/eoi:45500/componente45499.pdf

Torres, J. M. (s.f.). *SQL Server Compact 2008.*

8 8 ANEXOS

ANEXO 1: Manual De Usuario

Términos y Abreviaturas

Términos

Insumo.- Cualquier material y/o producto cuya finalidad es el mantenimiento del Consultorio o el servicio continúo a los pacientes.

Kardex.- Un reporte histórico de los movimientos: ingreso, egreso, anulación, transferencia o ajuste que se haya generado en un ítem que se encuentre en una bodega.

Venta.- Proceso mediante el cual un usuario ingresa al sistema informático y realiza uno o varios registros de medicamentos/insumos que ha sido utilizado en un paciente y/o procedimientos realizados al mismo para su venta.

Abreviaturas

CIE-10 – Código Internacional de Enfermedades

1. Objetivos

El presente manual tiene como objetivo proporcionar al personal del consultorio médico una herramienta y guía permanente para poder alimentar y gestionar toda la información que necesita ser ingresada al Sistema informático del Consultorio Salud & Vida.

2. Alcance/Limitaciones

El Sistema informático del Consultorio al momento ha presentado algunos afinamientos y cambios.

El presente manual se ha limitado exclusivamente a los procesos más importantes dentro de todo el sistema informático actual.

3. Generalidades

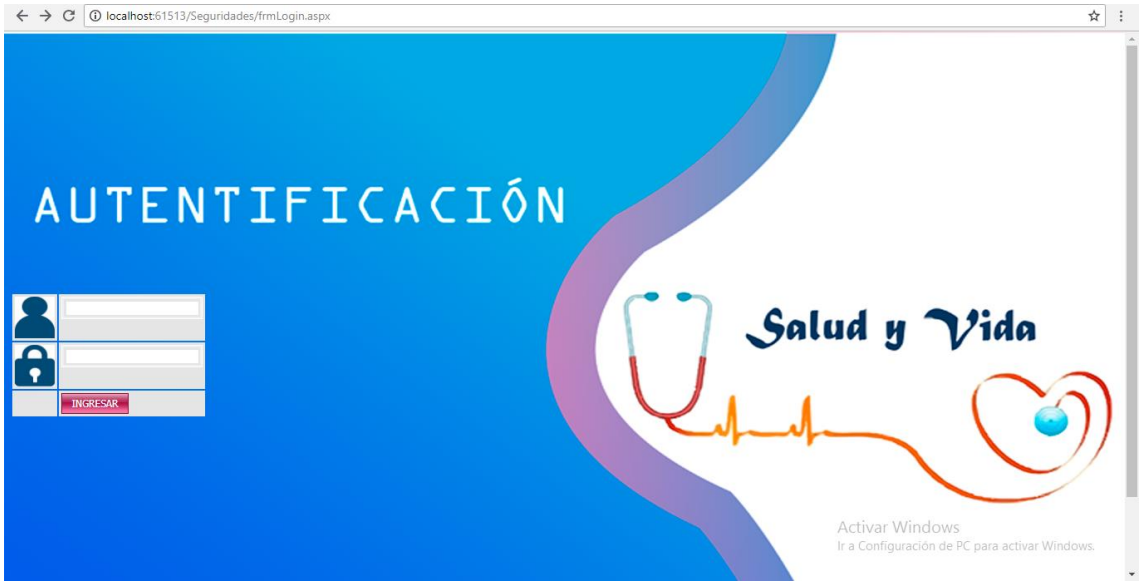
Toda la información que se ingresa al sistema informático, es tratada como información confidencial, por tal motivo, es necesario tener una auditoria y seguimiento de todas las transacciones que el usuario realiza en el mismo.

La responsabilidad de cada transacción realizada, es responsabilidad exclusiva del usuario que ha ingresado o modificado la información y que se encuentra registrado en el sistema. Por tanto es altamente recomendable que la clave de cada usuario sea una información confidencial y de uso exclusivo para cada colaborador.

4. Ingreso al Sistema

Para el ingreso al sistema se debe digitar la dirección:

http://localhost:61513/Seguridades/frmLogin.aspx



Dentro de esta pantalla es necesario ingresar el USUARIO y la CLAVE asignados a cada colaborador.

La bodega asignada al usuario vendrá **por defecto definida**.

5. Gestión de Pacientes

En este módulo el usuario autorizado, ingresa o modifica los datos del Paciente que forman la base fundamental para el registro de todos los consumos y/o servicios generados por el mismo. De tal manera que, siendo esta una información trascendental, debe darse la importancia y sumo cuidado en su registro.

5.1. Búsqueda y depuración de Pacientes

Para iniciar la gestión de un paciente, es imprescindible primero realizar la búsqueda del mismo seleccionando el paciente y presionando clic en el botón ***BUSCAR*** y/o **Seleccionar** El usuario puede hacer la búsqueda a través de los siguientes parámetros:

Paciente.- Apellidos y nombres del paciente. Mientras el usuario digite los caracteres el sistema le muestra un listado de coincidencias en este campo.

N° Identificación.- o número de documento que se haya registrado en el sistema.

Como resultado de la búsqueda el usuario visualizara la siguiente pantalla:

5.2. Actualización de Pacientes

Para continuar con el proceso, es indispensable que el usuario complete la siguiente información:

Datos.- Es indispensable registrar los datos que solicita el sistema.

Nro. Identificación.- el sistema no permitirá que continúe si existe algún error en el número de cédula:

Este dato es muy necesario y es altamente recomendable que se solicite al paciente o al familiar esta información ya que esto permite tener información correcta. **En casos excepcionales**, es decir por ejemplo en recién nacidos, el número de cédula que se ingresará será: 9999999999.

Teléfonos.- Esta información es un dato obligatorio para el registro de un nuevo paciente. Sin embargo en la actualización es necesario consultar al paciente si el teléfono que se encuentra registrado sigue siendo el mismo.

Email.- A pesar de que no es un dato obligatorio, el usuario debe actualizar esta información ya que es trascendental en las campañas publicitarias.

Cuando se han ingresado todos los datos, el usuario debe dar clic en **MODIFICAR** para guardar toda la información.

5.3. Ingreso de Pacientes

Antes de iniciar el proceso de ingreso de Pacientes, el usuario obligatoriamente debe hacer primero la búsqueda para que la información no se halle duplicada.

Para iniciar se debe hacer clic en el botón **NUEVO,** y se desplegará una pantalla con el formulario vacío para ingresar la información del paciente:

Es necesario tomar en cuenta todas las observaciones del punto “5.2 Actualización de Pacientes”, ya que estas mismas validaciones se realizarán al ingresar un paciente.

Adicionalmente, al ingresar un paciente, se valida que el número de cédula ingresado en este **PACIENTE NUEVO** no este registrado actualmente en el sistema.

Todos los datos del formulario deben ser ingresados por el usuario, a excepción del **Código** y la **Edad,** que son datos calculados por el sistema.

Luego de ingresados todos los datos requeridos, el usuario deberá dar clic en el botón ***GRABAR*** para finalizar la transacción.

6. Gestión de Productos/Servicios

6.1. Búsqueda y actualización de Productos

Los productos que están en la base de datos pueden ser buscados ubicándose en el nombre.

El sistema brinda la ayuda de búsqueda desplegando un listado con todas las coincidencias de los nombres encontrados, como se puede observar en la siguiente figura.

A seleccionar el producto, directamente nos muestra todos los datos del mismo:

En caso de requerir modificar algún dato, se debe tomar en cuenta las siguientes consideraciones:

- Los datos correspondientes a: Precio Compra y Precio Venta, deben ser números decimales. Si son decimales se debe ingresar el valor decimal con coma, ejemplo: [5,89].
- El control de inventario es indispensable que se seleccione en la opción **NO** únicamente cuando sea un servicio.
- El IVA debe seleccionar únicamente a los productos medicinas e insumos que se compre con IVA.

6.2. Ingreso de Productos

Este es un proceso que se ejecutará muy esporádicamente, solo cuando nuevos productos son adquiridos en el Consultorio. La base de datos de productos ya se encuentra alimentada y depurada en el sistema informático.

Se despliega una pantalla con todos los datos que el usuario debe ingresar para registrar un nuevo producto. Primero es necesario seleccionar que tipo de producto se va a ingresar entre las opciones están las listadas en el siguiente gráfico:

De la misma forma **y a manera de validación** para el usuario, al ingresar el **NOMBRE,** el sistema despliega un listado de los productos coincidentes con el que está digitando. De

esta forma el usuario podrá percatarse de que el medicamento está ingresado o no en la base de datos.

Considerar adicionalmente todas las validaciones.

7. Ingreso/Egreso de Inventario

Proceso mediante el cual el usuario autorizado ingresa al sistema todas las compras de medicamentos e insumos que se han generado. El ingreso de información de compras afecta directamente el Inventario.

7.1. Ingreso/Egreso

Se desplegará el formulario con los campos vacíos para su ingreso.

El usuario debe iniciar ingresando el **PROVEEDOR.** El sistema desplegará un listado de todas las coincidencias mientras el usuario digite en este campo, luego es necesario ingresar en número de documento el **Nro. Factura** de la compra.

El usuario debe seleccionar el producto a ser ingresado y posteriormente se visualizara la información del producto que ya existe en la base de datos:

El usuario deberá ingresar la cantidad adquirida, el precio de compra y hacer clic en **icono +.** Este proceso se lo debe realizar para cada tipo de producto adquirido.

La sumatoria del total de la compra la cual se calcula automáticamente deberá coincidir con el total de la factura física de la compra. Al final deberá grabar la información haciendo clic en el botón **GRABAR.**

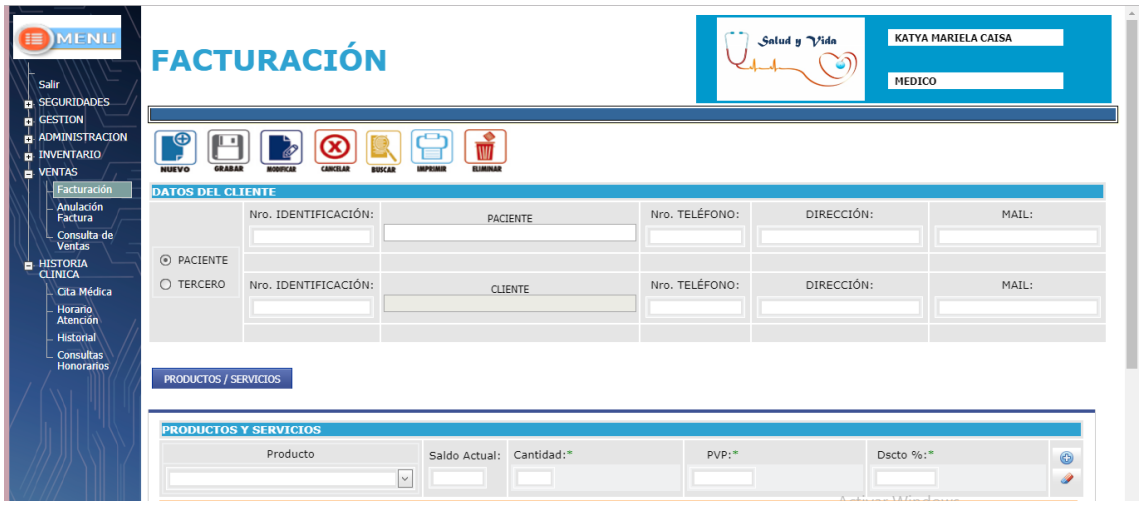
8. Gestión de Ventas

Mediante este proceso los usuarios autorizados podrán generar las facturas de ventas realizadas en el consultorio.

8.1. Ingreso de Ventas

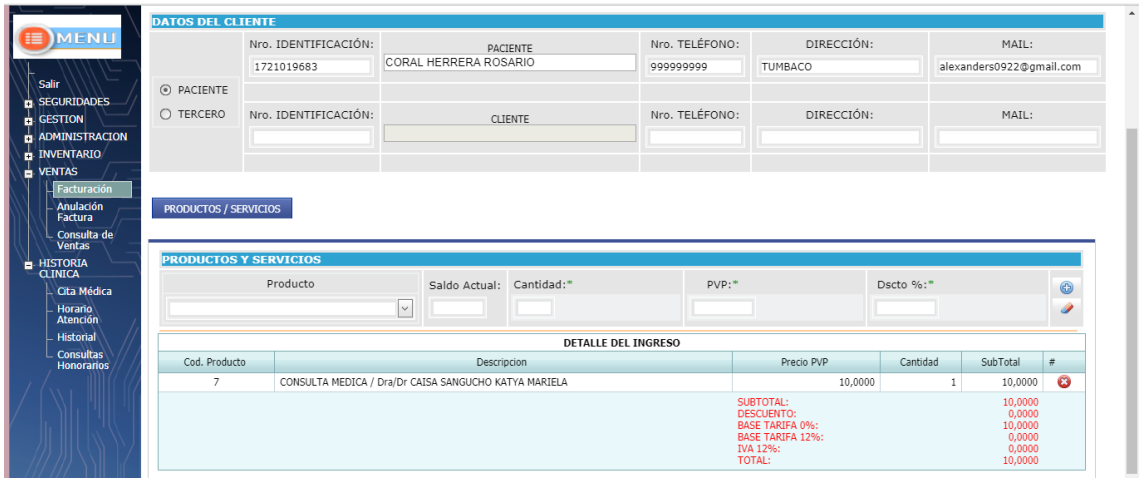
- PACIENTE.- cuando la factura se emitirá a nombre del paciente. Se habilita los campos para los datos del paciente.
- TERCEROS.- cuando la factura será emitida a nombre de un tercero y no del paciente. Se habilitan los campos para los datos del paciente y para otra Razón Social.

Para continuar con la generación de la factura, el usuario debe ingresar el apellido del paciente y seleccionar el requerido de la lista de coincidencias que el sistema despliega.



Luego se procede a ingresar los productos que se facturarán. El sistema brinda al usuario dos formas de buscar el producto, por su *NOMBRE*, solamente se digita el nombre en el

campo PRODUCTO, cuando se lo encuentra de la lista desplegada se selecciona el requerido y los demás datos son llenos en los campos respectivos automáticamente, luego ingresar la cantidad y hacer clic en **icono +.**



Luego de ingresar todos estos datos necesarios, el usuario deberá hacer clic en **Grabar.** La factura se grabará normalmente y se mostrará el siguiente mensaje:

Luego de grabar la venta, el usuario podrá observar el formato para realizar la impresión de la factura. Es estrictamente obligatorio que el usuario revise el número de la factura **FÍSICA** en la que va a imprimir y que coincida con el **NÚMERO DE FACTURA** que se encuentra en rojo en el formato que se va a imprimir:

8.2. Anulación de Venta

Cuando se ha ingresado mal los datos de una factura, u ocurrió un error en la impresión, es necesario que el usuario anule primeramente la factura para luego proceder a generar otra.

El usuario primero debe digitar el nombre del cliente a quien fue generada la factura, el sistema desplegará un listado de todas las facturas emitidas a este cliente. Seleccione el número de factura que se necesite anular y haga clic el **icono visto**.

Luego de seleccionar la factura, finalmente dar clic en **GRABAR,** para culminar la ANULACIÓN DE VENTA:

9. CITA MÉDICA

Mediante este proceso los usuarios autorizados podrán generar las citas médicas del consultorio.

9.1. Ingreso de Cita Médica

El usuario para reservar la cita médica primero debe seleccionar el médico y la fecha de la cita, automáticamente se mostrara los pacientes agendados y el horario de atención del médico.

Para continuar con el agendamiento el usuario debe seleccionar el paciente y la hora de atención y finalmente presionar el botón grabar para agendar su cita.

10. CONSULTA MÉDICA

Mediante este proceso el medico podrán generar la historia clínica de los pacientes que fueron agendados y por tanto se muestra la agenda del médico.

Caso 1: **Paciente Nuevo**

El médico para iniciar el registro de la Historia Clínica del paciente debe dar clic en el primer icono y le direccionara a la siguiente pantalla

Caso 2: **Paciente Existente**

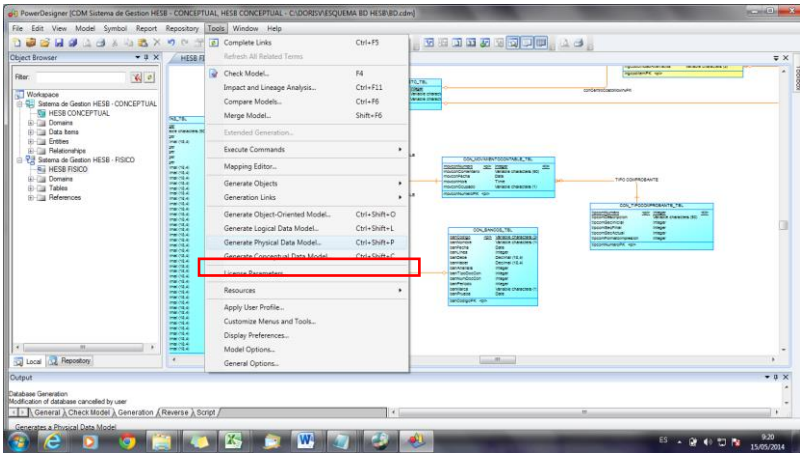
El médico para iniciar el registro de la Historia Clínica del paciente debe dar clic en el primer icono y le direccionara a la siguiente pantalla, en este caso como ya existe el historial del paciente recupera el mismo y lo muestra.

9 ANEXO 2: Manual Técnico

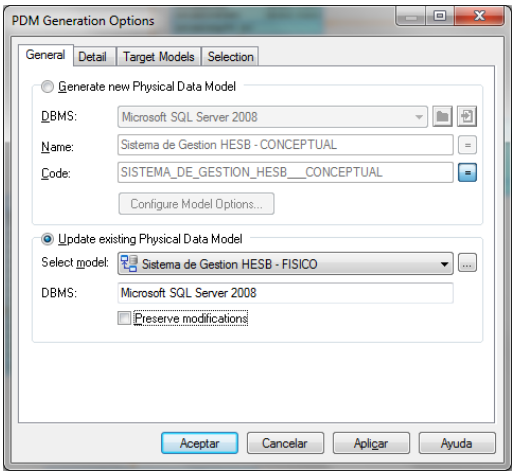
El desarrollo se encuentra realizado con las herramientas propias de Visual Studio 2010 C Sharp, UI DevExpress y SQL Server 2008R. Las herramientas utilizadas son para Diseño de la Interfaz y para el desarrollo el lenguaje de programación orientado a Objetos.

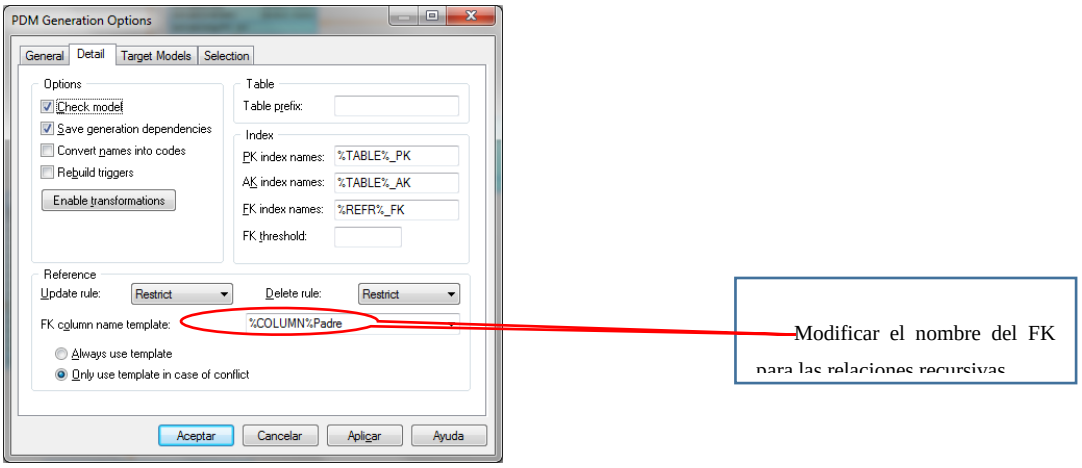
9.1 Generación de la Base de Datos desde Power Designer

1. Los cambios realizar en el modelo conceptual
2. Generar el modelo físico

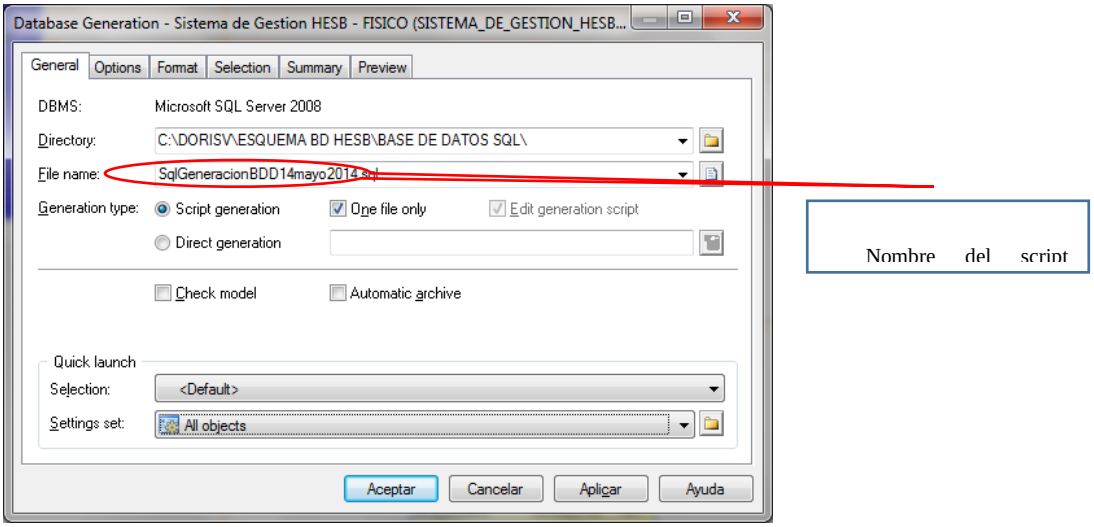


3. Actualizar el modelo físico en la siguiente pantalla:

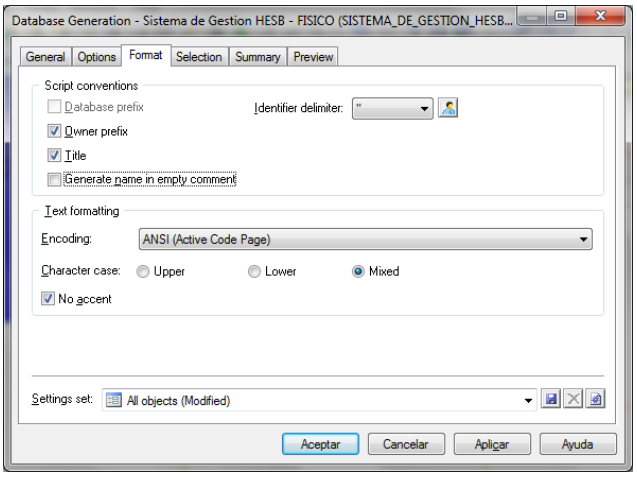




4. Para generar por primera vez la base de datos se debe definir las siguientes características:

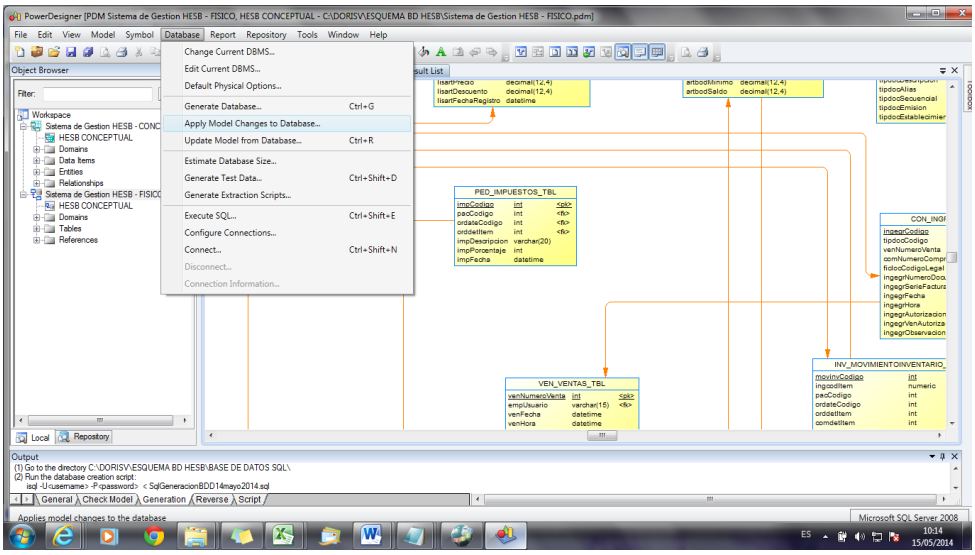


En la pestaña format quitar el generar el comentario en columnas que no lo tengan:

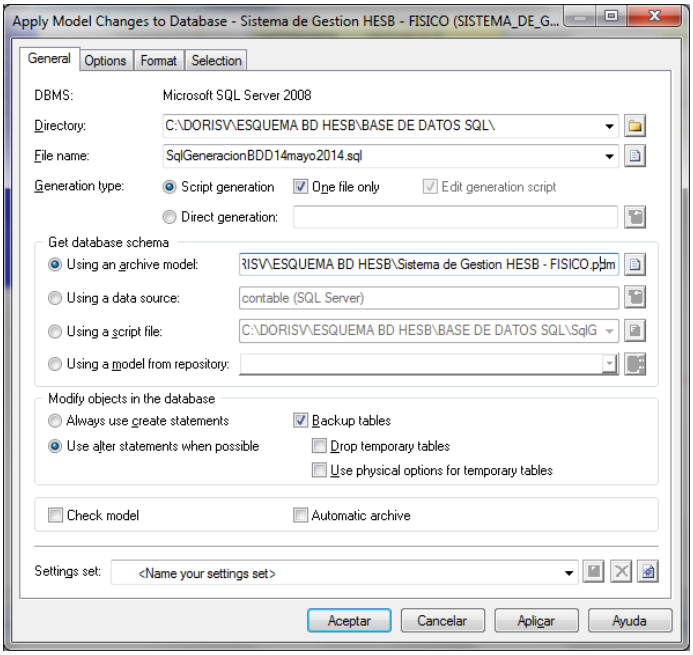


1.1 Para generar los cambios realizados en la base de datos se debe definir las siguientes características:

1. Seleccionar la opción Appy model changes to database:



2. Configurar como sigue:



De esta manera se generaran los **alter** necesarios para modificar la base de datos que ya se encuentra generada en el servidor.

En el campo “Using an archive model”, se debe seleccionar el modelo físico ultimo generado.

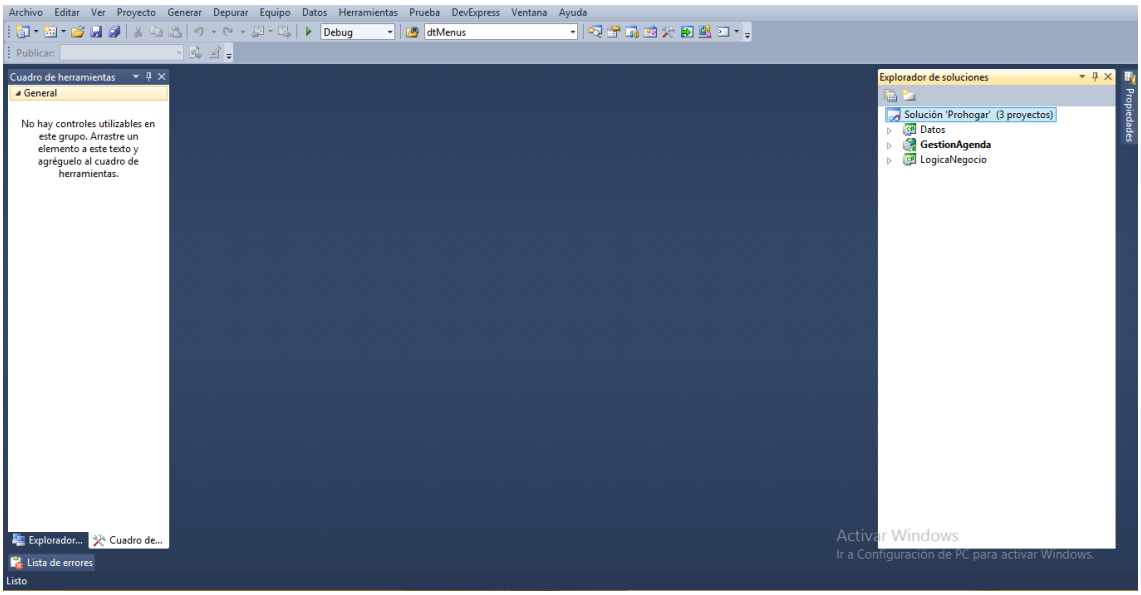
1.2 Para borrar la base de datos completamente

En la pestaña de OPTIONS se debe marcar todos los DROP de los objetos excepto el de la BDD.

Tener cuidado de poner en el nombre del script otro nombre para que no se reemplace el script con el que se creó.

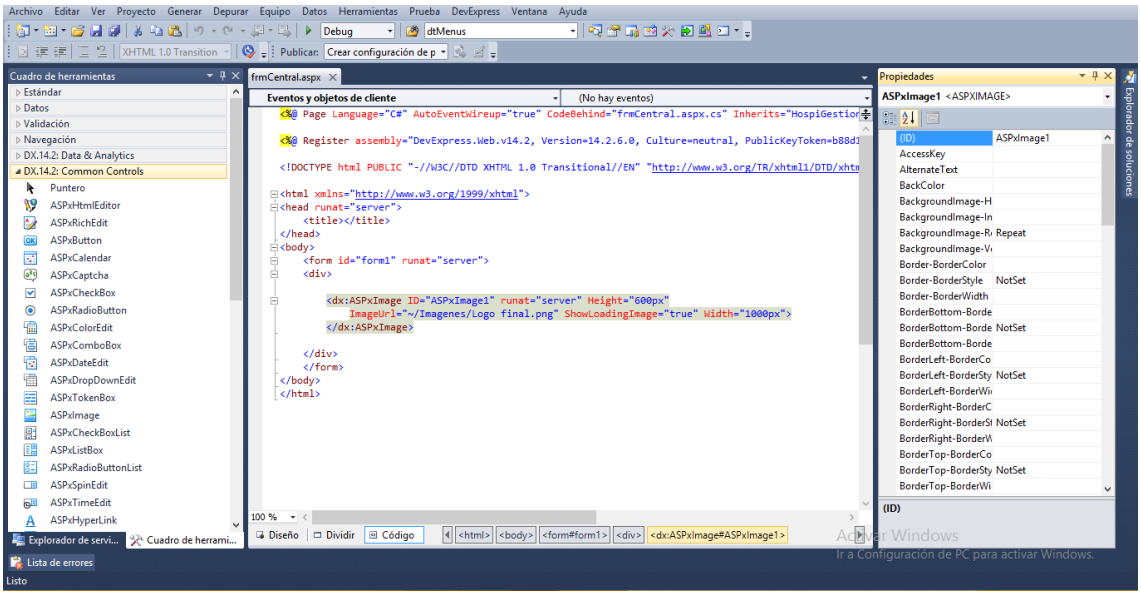
10 AMBIENTE DE DESARROLLO

Es el entorno en el cual el desarrollador puede manipular los objetos que son parte del Visual Studio 2010.

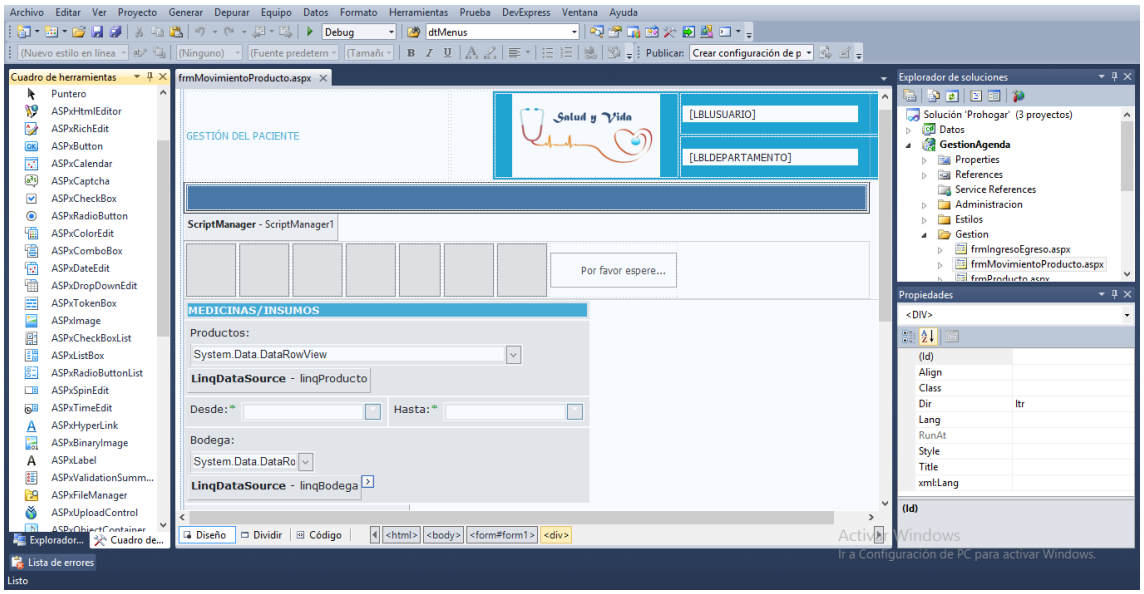


Proyecto

Es el conjunto de objetos que se encuentran agrupados por categoría para su administración técnica. Los objetos utilizados para este proyecto son: DevExpress

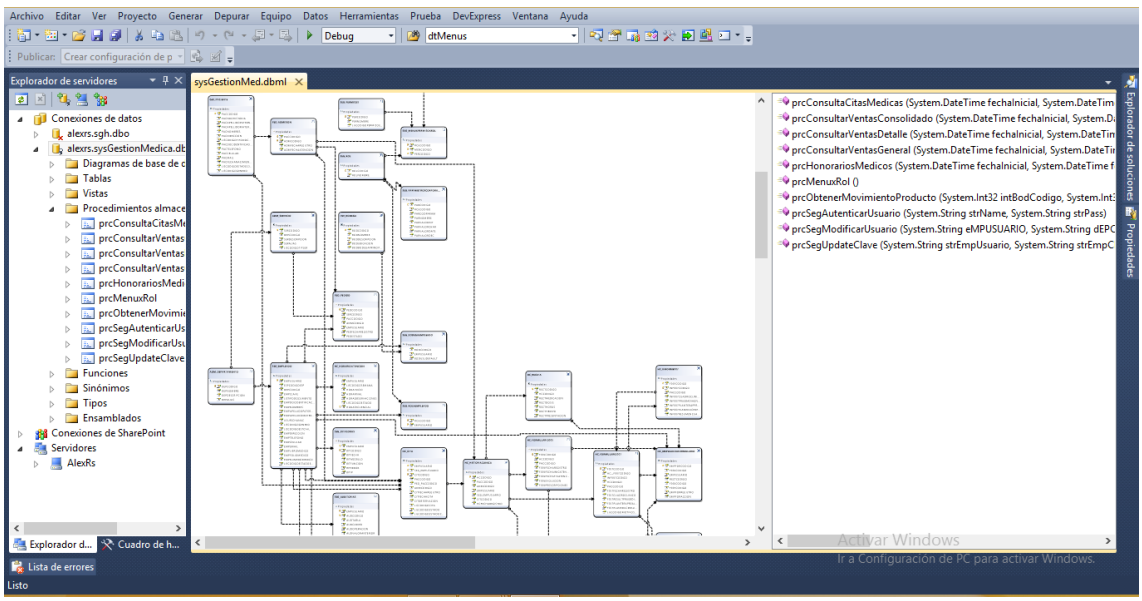


Ejemplo: Interfaz de Usuario



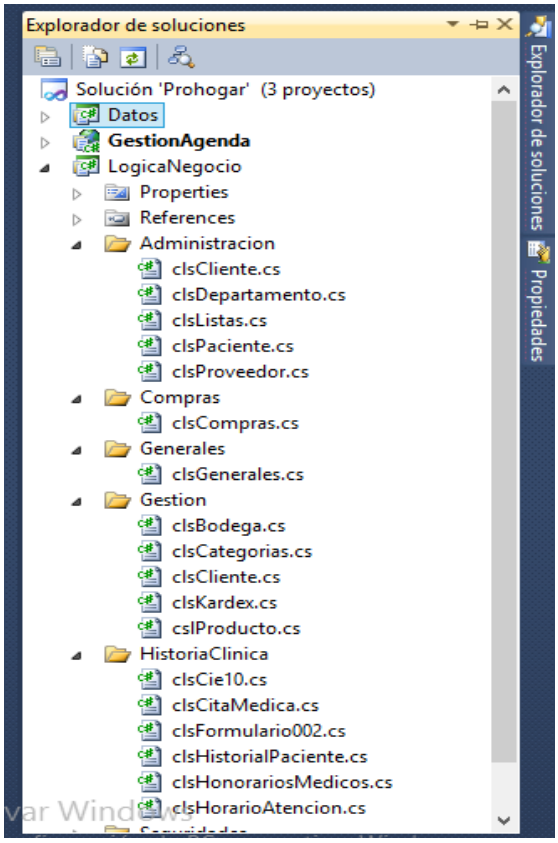
Tablas

Es la definición de cada estructura donde se almacenara la información, contiene campos, relaciones, índices, métodos.



Clases

Son objetos en los cuales se realiza la codificación de la solución, cada clase tiene sus métodos.



Componentes básicos de una página ASP.NET

En la primera línea destaca la directiva

```
<%@ Page Language="C#" %>
```

Que le indica al servidor Web que se trata de una página ASP.NET y que usa Visual Studio C# (C Sharp) como lenguaje de programación

Después de la directiva vienen bloques de declaración de código

```
<head runat="server"> </head>
<body>
        <div>
                <form id="form1" runat="server">
        </div>
                </form>
</body>
```

Aquí es donde debe ir todo o la mayor parte del código.

A lo largo de la página pueden aparecer elementos conocidos de HTML pero que se declaran de manera especial

```
<dx:ASPxLabel ID="lblFormulario" runat="server" CssClass="lblTitulo"
                Text="GESTIÓN DEL PACIENTE">
</dx:ASPxLabel>
```

Separación de Código C# y contenido HTML

Se puede mantener todo el código, tanto Visual Studio .NET como HTML, en un mismo archivo con extensión **aspx,** el código iría sobre todo en bloques de declaración de código (en la cabecera del documento) y el código HTML iría en el cuerpo del documento.

Programación del lado del cliente: JavaScript

JavaScript permite crear aplicaciones específicamente orientada a su funcionamiento en la red Internet

```
<script language="JavaScript" type="text/javascript">
        function funOnClickDateEdit(s, e) {
            ASPxClientUtils.AttachEventToElement(s.GetInputElement(), "click",
function (event) {
                s.ShowDropDown();
            });
        }
  </script>
```

Clase de inicio de Sesión

Definición de la clase y sus variables globales

10.1.1 Variables de Session

Al ingresar el usuario, se configurarán en sesión las siguientes variables:

- `(string)Session["strUsuario"]`

  Almacena el username del usuario para ser usado en cualquier página del sistema cuando el usuario ya está logeado en la aplicación.
- `(string)Session["strNombreUsuario"]`

Almacena el nombre y apellido del usuario para ser desplegado en cualquier página del sistema cuando el usuario ya está logeado en la aplicación.

- `DataTable dtRoles = Session["dttMenues"]`

Almacena los menues, permisos y roles que tiene asignados el usuario para ser desplegados o usados en cualquier página del sistema cuando el usuario ya está logeado en la aplicación. Esta variable de sesión se la deberá transformar a un datatable para poder recorrer todos los registros almacenados en la variable.

Ejemplo clsSession

```
public class clsSession

    {
        //Constantes de variables de sesión
        public const string strNombrePaciente = "sesNombrePaciente";
        public const string strUsuario = "sesUsuario";
        public const string strNombreUsuario = "sesNombreUsuario";
        public const string strDepartamento = "sesDepartamento";
        public const string strDepartamentoCodigo = "sesDepartamentoCodigo";
        public const string strIP = "sesIP";
        public const string strPuntoVenta = "sesPuntoVenta";
        public const string strRoles = "dttRoles";
        public const string strMenus = "ddtMenues";

        public static T prcLee<T>(string variable)
        {
            object valor = HttpContext.Current.Session[variable];
            if (valor == null)
                return default(T);
            else
                return ((T)valor);
        }
```

```
public static void prcEscribe(string variable, object valor)
{
    HttpContext.Current.Session[variable] = valor;
}


public static string sesNombrePaciente
{
    get
    {
        return prcLee<string>(strNombrePaciente);
    }
    set
    {
        prcEscribe(strNombrePaciente, value);
    }
}

public static string sesUsuario
{
    get
    {
        return prcLee<string>(strUsuario);
    }
    set
    {
        prcEscribe(strUsuario, value);
    }
}

public static string sesNombreUsuario
{
    get
    {
        return prcLee<string>(strNombreUsuario);
    }
    set
    {
        prcEscribe(strNombreUsuario, value);
    }
}

public static string sesDepartamento
{
    get
    {
        return prcLee<string>(strDepartamento);
    }
    set
    {
        prcEscribe(strDepartamento, value);
    }
}

public static string sesIdDepartamento
{
    get
    {
        return prcLee<string>(strDepartamentoCodigo);
    }
    set
    {
        prcEscribe(strDepartamentoCodigo, value);
    }
}

public static string sesIP
```

```
{
    get
    {
        return prcLee<string>(strIP);
    }
    set
    {
        prcEscribe(strIP, value);
    }
}

public static string sesPuntoVenta
{
    get
    {
        return prcLee<string>(strPuntoVenta);
    }
    set
    {
        prcEscribe(strPuntoVenta, value);
    }
}


public static DataTable sesRoles
{
    get
    {
        return prcLee<DataTable>(strRoles);
    }
    set
    {
        prcEscribe(strRoles, value);
    }
}

public static DataTable sesMenus
{
    get
    {
        return prcLee<DataTable>(strMenus);
    }
    set
    {
        prcEscribe(strMenus, value);
    }
}
```

10.1.2 Clases del Giro del Negocio

Ejemplo clsHistoriaClinica

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Datos;
using System.Data.Linq;
using System.Data.Linq.SqlClient;

namespace LogicaNegocio.HistoriaClinica
{
    public class clsHistorialPaciente
    {
        public static Datos.sysGestionMedDataContext dc;
        private static string strContexto = " DECLARE @context_info arbinary(100);" +
                                            " SET @context_info = cast('strUserIP' as
                                                varbinary(100));" +
                                            " SET CONTEXT_INFO @context_info; ";
        //Define y retorna la estructura de la tabla
        public static HC_HISTORIACLINICA prcTablHistorialHC()
        {
            HC_HISTORIACLINICA Lista = new HC_HISTORIACLINICA();
            return Lista;
        }
        public static List<HC_HISTORIACLINICA> prcObtenerListHistorialPaciente(int intCodigo)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            var lista = dc.HC_HISTORIACLINICA.Where(row => row.CITCODIGO.Equals(intCodigo));
            return lista.ToList();
        }
        //Inserta la HC_HISTORIACLINICA_TBL y su respectiva auditoria
        public static void prcInsert(HC_HISTORIACLINICA InfoHistorialHC, string strUserIP)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            try
            {
                dc.Connection.Open();
                dc.Transaction = dc.Connection.BeginTransaction();

                //Envia los datos de usuario e IP a setear en el CONTEXT_INFO para grabar en la
auditoria
                dc.ExecuteCommand(strContexto.Replace("strUserIP", strUserIP));

                //Inserta la Admision del paciente en el Historial de Historia Clinica del paciente
                InfoHistorialHC.HCFECHAREGISTRO = System.DateTime.Now;
                dc.HC_HISTORIACLINICA.InsertOnSubmit(InfoHistorialHC);
                dc.SubmitChanges();
                dc.Transaction.Commit();
            }
            catch (SystemException ex)
            {
                dc.Transaction.Rollback();
LogicaNegocio.Generales.clsGenerales.prcRegistraBitacora("logicaNegocio.HistoriaClinica.clsHistoria
lHistoriaClinica", "prcInsert()", ex, strUserIP);
                throw new ArgumentException("ERROR: Datos no guardados. " + ex.Message.Replace("\"",
"").Replace(Char.ConvertFromUtf32(13), " ").Replace(Char.ConvertFromUtf32(10), ""));
            }
            finally
            {
                dc.Connection.Close();
                dc.Transaction = null;
            }
        }
    }
}
```

Ejemplo clsVentas

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Datos;
using System.Data.Linq.SqlClient;

namespace LogicaNegocio.Ventas
{
    public class clsVentas
    {
        public static Datos.sysGestionMedDataContext dc;
        private static string strContexto = " DECLARE @context_info varbinary(100);" +
                                    " SET @context_info = cast('strUserIP' as varbinary(100));" +
                                    " SET CONTEXT_INFO @context_info; ";
        //Define y retorna la estructura de la tabla LISTA
        public static VEN_VENTAS prcTablaVentas()
        {
            VEN_VENTAS lisLista = new VEN_VENTAS();
            return lisLista;
        }
        //Obtiene el detalle de las ventas
        public static List<dynamic> funObtenerVentasxDetalle(DateTime datFechaInicial, DateTime
datFechaFinal, string strPacienteRazonSocial, string strTipoPacienteRazonSocial, Int64
intNumeroFactura)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            var lisDetalles = dc.prcConsultarVentasDetalle(datFechaInicial, datFechaFinal,
strPacienteRazonSocial, strTipoPacienteRazonSocial, intNumeroFactura);
            return lisDetalles.ToList<dynamic>();
        }
        //Obtiene el detalle de las ventas
        public static List<dynamic> funObtenerVentasxConsolidado(DateTime datFechaInicial, DateTime
datFechaFinal, string strPacienteRazonSocial, string strTipoPacienteRazonSocial, Int64
intNumeroFactura)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            var lisDetalles = dc.prcConsultarVentasConsolidado(datFechaInicial, datFechaFinal,
strPacienteRazonSocial, strTipoPacienteRazonSocial, intNumeroFactura);
            return lisDetalles.ToList<dynamic>();
        }
        //Obtiene los pacientes/terceros que tienen facturas
        public static List<string> prcObtenerPacientesTerceros(string strTexto, int intCuantos)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            var listaPacientes = (from PAC in dc.GES_PACIENTE.DefaultIfEmpty()
                                  join ven in dc.VEN_VENTAS.DefaultIfEmpty() on PAC.PACCODIGO equals
ven.PACCODIGO
                                  where SqlMethods.Like((PAC.PACAPELLIDOPATERNO ?? "") + " " +
(PAC.PACAPELLIDOMATERNO ?? "") + " " + (PAC.PACNOMBRES ?? ""), strTexto.Replace(" ", "%") + "%")
                                  select ((PAC.PACAPELLIDOPATERNO ?? "") + ' ' +
(PAC.PACAPELLIDOMATERNO ?? "") + ' ' + (PAC.PACNOMBRES ?? "") + " |" +
PAC.PACDOCIDENTIFICACION)).Distinct();

            var listaFichas = (from cli in dc.ADM_CLIENTE.DefaultIfEmpty()
                               join ven in dc.VEN_VENTAS.DefaultIfEmpty() on cli.CLICODIGO equals
ven.CLICODIGO
                               where SqlMethods.Like((cli.CLINOMBRE ?? ""), strTexto.Replace(" ",
"%") + "%")
                               select ((cli.CLINOMBRE ?? "") + " |" + cli.CLICODIGO)).Distinct();
            return listaPacientes.Concat(listaFichas).Take(intCuantos).ToList();
        }
        public static Int16 prcInsert(VEN_VENTAS InfoIngresoProductos, string[,] arrIngresoDetalles,
string strUserIP)
```

```
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            try
            {
                decimal decSubtotal = 0;
                decimal decTotalVenta = 0;
                decimal decTotalConIva = 0;
                decimal decTotalSinIva = 0;
                decimal decIva12 = 0;
                decimal decDscto = 0;
                dc.Connection.Open();
                dc.Transaction = dc.Connection.BeginTransaction();
                //Envía los datos de usuario e IP a setear en el CONTEXT_INFO para grabar en la
                auditoria
                dc.ExecuteCommand(strContexto.Replace("strUserIP", strUserIP));
                //Bloquea la tabla pedido y kardex para que no se inserte el mismo código
                nuevamente (lock)
                dc.ExecuteCommand("select max(procodigo) from ven_ventadetalle WITH (TABLOCK,
                HOLDLOCK)");
                dc.ExecuteCommand("select max(karcodigo) from inv_kardex WITH (TABLOCK,
                HOLDLOCK)");
                dc.ExecuteCommand("select max(procodigo) from inv_Producto WITH (TABLOCK,
                HOLDLOCK)");
                //Inserta el INGRESO
                InfoIngresoProductos.VENCODIGO = dc.VEN_VENTAS.Any() ? dc.VEN_VENTAS.Max(row =>
                row.VENCODIGO) + 1 : 1;
                dc.VEN_VENTAS.InsertOnSubmit(InfoIngresoProductos);
                //Inserta detalles en INV_INGRESOSINVENTARIODET_TBL
                for (int i = 0; i <= ((arrIngresoDetalles.Length / 15) - 1); i++)
                {
                    VEN_VENTADETALLE InfoIngresoProductosDetalles = new VEN_VENTADETALLE();
                    InfoIngresoProductosDetalles.PROCODIGO  =  Convert.ToInt32(arrIngresoDetalles[i,
                        0].ToString());
                    InfoIngresoProductosDetalles.CATCODIGO  =  Convert.ToInt32(arrIngresoDetalles[i,
                        13].ToString());
                    InfoIngresoProductosDetalles.PXPCODIGO  =  Convert.ToInt32(arrIngresoDetalles[i,
                        14].ToString());
                    InfoIngresoProductosDetalles.VENCODIGO = InfoIngresoProductos.VENCODIGO;
                    InfoIngresoProductosDetalles.DVENDESCRIPCION = arrIngresoDetalles[i,
                        1].ToString();
                    InfoIngresoProductosDetalles.DVENCANTIDAD =
                        Convert.ToInt32(arrIngresoDetalles[i, 3].ToString());
                    InfoIngresoProductosDetalles.DVENPVP  =  Convert.ToDecimal(arrIngresoDetalles[i,
                        2].ToString());
                    InfoIngresoProductosDetalles.DVENPRECIOCOMPRA =
                        Convert.ToDecimal(arrIngresoDetalles[i, 4].ToString());
                    InfoIngresoProductosDetalles.DVENPRECIOVENTA =
                        Convert.ToDecimal(arrIngresoDetalles[i, 5].ToString()) /
                        Convert.ToInt32(arrIngresoDetalles[i, 3].ToString());
                    InfoIngresoProductosDetalles.DVENIVAVALOR =
                        Convert.ToDecimal(arrIngresoDetalles[i, 9].ToString());
                    InfoIngresoProductosDetalles.DVENDESCUENTOPORCENTAJE =
                        Convert.ToDecimal(arrIngresoDetalles[i, 10].ToString());
                    InfoIngresoProductosDetalles.DVENDESCUNETOVALOR =
                        Convert.ToDecimal(arrIngresoDetalles[i, 6].ToString());
                    InfoIngresoProductosDetalles.DVENSUBTOTAL =
                        Convert.ToDecimal(arrIngresoDetalles[i, 5].ToString());
                    InfoIngresoProductosDetalles.DVENTARIFA0 =
                        Convert.ToDecimal(arrIngresoDetalles[i, 8].ToString());
                    InfoIngresoProductosDetalles.DVENTARIFA12 =
                        Convert.ToDecimal(arrIngresoDetalles[i, 7].ToString());
                    InfoIngresoProductosDetalles.EMPMEDICO = arrIngresoDetalles[i, 11].ToString();
                    // SubTotales
                    decSubtotal    =    decSubtotal    +    Convert.ToDecimal(arrIngresoDetalles[i,
5].ToString());
                    decDscto = decDscto + Convert.ToDecimal(arrIngresoDetalles[i, 6].ToString());
                    decTotalConIva   =   decTotalConIva   +   Convert.ToDecimal(arrIngresoDetalles[i,
7].ToString());
                    decTotalSinIva   =   decTotalSinIva   +   Convert.ToDecimal(arrIngresoDetalles[i,
8].ToString());
```

```
                    decIva12 = decIva12 + Convert.ToDecimal(arrIngresoDetalles[i, 9].ToString());
                    decTotalVenta = (decSubtotal - decDscto) + decIva12;
                    InfoIngresoProductosDetalles.DVENTOTALVENTA                                    =
Convert.ToDecimal(decTotalVenta.ToString());
                    dc.VEN_VENTADETALLE.InsertOnSubmit(InfoIngresoProductosDetalles);
                    //Ingresa el registro en el kardex
                    //SI ES UN PRODUCTO QUE NECESITA LLEVAR INVENTARIO
                    string strInventario = "N";
                    strInventario                                                                  =
(LogicaNegocio.Gestion.cslProducto.prcObtenerListProductooxCodigo(Convert.ToInt32(arrIngresoDetalle
s[i, 0].ToString())))[0].PROINVENTARIO.ToString();
                    if (strInventario == "S")                    {
                        //Modifica el saldo de la bodega por default
                        int intBodega = Convert.ToInt16(arrIngresoDetalles[i, 12].ToString());
                        dc.ExecuteCommand("select PROSTOCK from INV_PRODUCTOXBODEGA  WITH (TABLOCK,
HOLDLOCK)   where   bodcodigo="   +   intBodega.ToString()   +   "   and   procodigo   =   "   +
InfoIngresoProductosDetalles.PROCODIGO);
                        //Actualiza el saldo en las bodegas
                        var      obj      =      dc.INV_PRODUCTOXBODEGA.SingleOrDefault(row      =>
row.PROCODIGO.Equals(InfoIngresoProductosDetalles.PROCODIGO)
                                                                                        &&
row.BODCODIGO.Equals(intBodega));
                        //si el ajuste es en menos, resta
                        obj.PROSTOCK   =   obj.PROSTOCK   -   Convert.ToInt32(arrIngresoDetalles[i,
3].ToString());
                        int intSaldo = Convert.ToInt32(obj.PROSTOCK);
                        //Inserta el movimiento en el Kardex
                        INV_KARDEX InfoKardex = new INV_KARDEX();
                        InfoKardex.KARCODIGO   =   dc.INV_KARDEX.Any()   ?   dc.INV_KARDEX.Max(row   =>
row.KARCODIGO) + 1 + i : 1 + i;
                        InfoKardex.TIPDOCCODIGO                                                    =
Convert.ToInt32(InfoIngresoProductos.TIPDOCCODIGO);
                        InfoKardex.BODCODIGO = intBodega;
                        InfoKardex.PROCODIGO                                                       =
Convert.ToInt32(InfoIngresoProductosDetalles.PROCODIGO);
                        InfoKardex.KARNUMEROTRANSACCION                                            =
Convert.ToInt32(InfoIngresoProductos.VENCODIGO);
                        InfoKardex.KARSALDO = intSaldo;
                        InfoKardex.KARCANTIDAD        =        Convert.ToInt32(arrIngresoDetalles[i,
3].ToString());
                        InfoKardex.KARFECHATRANSACCION = System.DateTime.Now;
                        dc.INV_KARDEX.InsertOnSubmit(InfoKardex);
                    }
                }
                ////
                dc.SubmitChanges();
                dc.Transaction.Commit();
                LogicaNegocio.Ventas.clsVentas.prcUpdateVenta(decSubtotal, decDscto, decTotalConIva,
decTotalSinIva, decIva12, decTotalVenta, InfoIngresoProductos, dc);
                return Convert.ToInt16(InfoIngresoProductos.VENCODIGO);
            }
            catch (SystemException ex)
            {
                dc.Transaction.Rollback();

LogicaNegocio.Generales.clsGenerales.prcRegistraBitacora("logicaNegocio.Compras.clsCompras.cs",
"prcInsert()", ex, strUserIP);
                throw new ArgumentException("ERROR: Datos no guardados. " + ex.Message.Replace("\"",
"").Replace(Char.ConvertFromUtf32(13), " ").Replace(Char.ConvertFromUtf32(10), ""));
            }
            finally
            {
                dc.Connection.Close();
                dc.Transaction = null;
            }
        }
        public   static   void   prcUpdateVenta(decimal   decSubtotal,   decimal   decDscto,   decimal
decTotalConIva, decimal decTotalSinIva,
                                        decimal   decIva12,   decimal   decTotalVenta,   VEN_VENTAS
InfoIngresoProductos, sysGestionMedDataContext dc)
```

```
        {
            try
            {
                // Realiza la actualizacion de la cita con su respectivo motivo de anulacion y registra
el ultimo estado
                dc.ExecuteCommand("UPDATE VEN_VENTAS SET DVENSUBTOTAL = {0} , DVENTOTALDSCTO =  {1}
, DVENTARIFA0 = {2}, DVENTARIFA12 = {3}, DVENTOTALIVA = {4}, DVENTOTALVENTA = {5}   WHERE VENCODIGO
= {6}", new object[]
                    {
                        decSubtotal,
                        decDscto,
                        decTotalSinIva,
                        decTotalConIva,
                        decIva12,
                        decTotalVenta,
                        InfoIngresoProductos.VENCODIGO
                    });
                dc.SubmitChanges();
                //  dc.Transaction.Commit();
            }
            catch (Exception ex)
            {
                dc.Transaction.Rollback();
                throw    new    ArgumentException("ERROR:    Datos    no    modificados.    "    +
ex.Message.Replace("\"",                "").Replace(Char.ConvertFromUtf32(13),                "
").Replace(Char.ConvertFromUtf32(10), ""));
            }
            finally
            {
                dc.Connection.Close();
                dc.Transaction = null;
            }
        }
        public static void prcUpdateVentaDetalle(Int64 intVenDetItem, string strUserIP)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            try
            {
                dc.Connection.Open();
                dc.Transaction = dc.Connection.BeginTransaction();
                //Envia los datos de usuario e IP a setear en el CONTEXT_INFO para grabar en la
auditoria
                dc.ExecuteCommand(strContexto.Replace("strUserIP", strUserIP));
                var   regVenDetItem   =   dc.VEN_VENTADETALLE.Where(r   =>   r.VENNUMEROLINEA   ==
intVenDetItem).FirstOrDefault();
                regVenDetItem.DVENHONORARIOPAGADO = "SI";
                dc.SubmitChanges();
                dc.Transaction.Commit();
            }
            catch (SystemException ex)
            {
                dc.Transaction.Rollback();

LogicaNegocio.Generales.clsGenerales.prcRegistraBitacora("logicaNegocio.Ventas.clsVentaDetalle",
"prcUpdateVentaDetalle()", ex, strUserIP);
                throw new ArgumentException("ERROR: Datos no guardados. " + ex.Message.Replace("\"",
"").Replace(Char.ConvertFromUtf32(13), " ").Replace(Char.ConvertFromUtf32(10), ""));
            }
            finally
            {
                dc.Connection.Close();
                dc.Transaction = null;
            }
        }

    }
}
```

Ejemplo clsProducto

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Datos;
using System.Data.Linq;
using System.Data.Linq.SqlClient;

namespace LogicaNegocio.Gestion
{
    public class cslProducto
    {
        public static Datos.sysGestionMedDataContext dc;
        private static string strContexto = " DECLARE @context_info varbinary(100);" +
                                            "    SET    @context_info    =    cast('strUserIP'    as
varbinary(100));" +
                                            " SET CONTEXT_INFO @context_info; ";

        //Define y retorna la estructura de la tabla
        public static INV_PRODUCTO prcTablaProducto()
        {
            INV_PRODUCTO Lista = new INV_PRODUCTO();
            return Lista;
        }
        //Define y retorna la estructura de la tabla
        public static INV_PRECIOXPRODUCTO prcTablaPrecioxProducto()
        {
            INV_PRECIOXPRODUCTO Lista = new INV_PRECIOXPRODUCTO();
            return Lista;
        }

        public static List<string> prcBuscarProducto(string strTexto, int intCuantos)
        {
            sysGestionMedDataContext dc = new sysGestionMedDataContext();
            var lista = (from c in dc.INV_PRODUCTO
```

```
                        where SqlMethods.Like((c.PRONOMBRECOMERCIAL ?? ""), strTexto.Replace(" ",
"%") + "%")
                        select ((c.PRONOMBRECOMERCIAL ?? ""))).Distinct();
            return lista.Take(intCuantos).ToList();
        }

        public static List<INV_PRODUCTO> prcBuscarProductoxNombre(string strNombreProducto)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            var lista = from c in dc.INV_PRODUCTO
                        where c.PRONOMBRECOMERCIAL == strNombreProducto
                        select c;
            return lista.ToList();
        }

        //Obtiene el codigo del precio de un producto
        public static List<INV_PRECIOXPRODUCTO> prcObtenerPrecioReal(Int32 intproCodigo)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            List<INV_PRECIOXPRODUCTO> lisPrecio = new List<INV_PRECIOXPRODUCTO>();

            var list = (from pre in dc.INV_PRECIOXPRODUCTO.DefaultIfEmpty()
                        where pre.PROCODIGO == intproCodigo
                        && pre.LISCODIGOESTADOPRECIO == "ACT"
                        select pre);
            lisPrecio = list.ToList();
            return lisPrecio;
        }
        //Obtiene el codigo del precio de un producto
        public static List<INV_PRODUCTOXBODEGA> prcObtenerSaldoProductoxBodega(Int32 intCodProducto,
int intBodCodigo)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            var lista = (from PXB in dc.INV_PRODUCTOXBODEGA
                         where PXB.PROCODIGO == intCodProducto
                         && PXB.BODCODIGO == intBodCodigo
                         select PXB);
            return lista.ToList();
        }


        //Obtiene los articulos de una categoria especifica y de un tipo especifico
        public static List<dynamic> prcObtenerProductosxCategorias()
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            var lista = (from PRO in dc.INV_PRODUCTO.DefaultIfEmpty()
                         select new
                         {
                             PRO.PROCODIGO,
                             PRO.PRONOMBRECOMERCIAL
                         }).OrderBy(r => r.PRONOMBRECOMERCIAL);
            return lista.ToList<dynamic>();
        }

        //Obtiene los articulos de una categoria especifica y de un tipo especifico
        public static List<dynamic> prcObtenerProductosxCategoriasInventario()
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            var lista = (from PRO in dc.INV_PRODUCTO.DefaultIfEmpty()
                         where PRO.CATCODIGO <= 2
                         select new
                         {
                             PRO.PROCODIGO,
                             PRO.PRONOMBRECOMERCIAL
                         }).OrderBy(r => r.PRONOMBRECOMERCIAL);
            return lista.ToList<dynamic>();
        }

        public static List<INV_PRODUCTO> prcObtenerListProductooxCodigo(Int32 intCodigo)
        {
```

```
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            var lista = from c in dc.INV_PRODUCTO
                        where SqlMethods.Equals(c.PROCODIGO, intCodigo)
                        select c;
            return lista.ToList();
        }

        //INSERT PRODUCTO
        public static void prcInsertProducto(INV_PRODUCTO InfoProducto, INV_PRECIOXPRODUCTO
InfoPrecioxProducto, string[,] arrMaxProducto, string strUserIP)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            try
            {
                dc.Connection.Open();
                dc.Transaction = dc.Connection.BeginTransaction();
                //Envia los datos de usuario e IP a setear en el CONTEXT_INFO para grabar en la
auditoria
                dc.ExecuteCommand(strContexto.Replace("strUserIP", strUserIP));
                //Inserta la informacion del producto
                InfoProducto.PROCODIGO = dc.INV_PRODUCTO.Any() ? dc.INV_PRODUCTO.Max(row =>
row.PROCODIGO) + 1 : 1;
                dc.INV_PRODUCTO.InsertOnSubmit(InfoProducto);

                //Guarda el codigo del articulo para el retorno
                Int32 intCodigoArticuloRetorno = Convert.ToInt32(InfoProducto.PROCODIGO);
                //Inserta la informacion del PrecioxProducto
                InfoPrecioxProducto.PROCODIGO =
Convert.ToInt32(intCodigoArticuloRetorno.ToString());
                InfoPrecioxProducto.PXPCODIGO = dc.INV_PRECIOXPRODUCTO.Any() ?
dc.INV_PRECIOXPRODUCTO.Max(row => row.PXPCODIGO) + 1 : 1;
                InfoPrecioxProducto.PXPFECHAREGISTRO = System.DateTime.Now;
                InfoPrecioxProducto.PXPFECHAVIGENCIADESDE = System.DateTime.Now;
                dc.INV_PRECIOXPRODUCTO.InsertOnSubmit(InfoPrecioxProducto);
               //Inserta la informacion de ARTICULOXBODEGA
                bool boolIngresaArticuloxBodega = false;

                if (arrMaxProducto != null)
                {
                    for (int i = 0; i <= (arrMaxProducto.Length / 2) - 1; i++)
                    {
                        if (arrMaxProducto[i, 0] != null)
                        {
                            INV_PRODUCTOXBODEGA InfoProductoxBodega = new INV_PRODUCTOXBODEGA();
                            InfoProductoxBodega.PROCODIGO = Convert.ToInt32(InfoProducto.PROCODIGO);
                            InfoProductoxBodega.BODCODIGO = Convert.ToInt32(arrMaxProducto[i,
0].ToString());
                            InfoProductoxBodega.PROMAXIMO = Convert.ToDecimal(arrMaxProducto[i,
1].ToString());
                            InfoProductoxBodega.PROMINIMO = 0;
                            dc.INV_PRODUCTOXBODEGA.InsertOnSubmit(InfoProductoxBodega);
                            boolIngresaArticuloxBodega = true;
                        }
                    }
                }
                //Inserta un registro en 0 para que no se genere error en las compras
                //Si no ingreso un registro en articulo xbodega y si es un producto (no servicio)
                if (boolIngresaArticuloxBodega == false && arrMaxProducto != null)
                {
                    INV_PRODUCTOXBODEGA InfoArticuloxBodegaNuevo = new INV_PRODUCTOXBODEGA();
                    InfoArticuloxBodegaNuevo.PROCODIGO = Convert.ToInt32(InfoProducto.PROCODIGO);
                    InfoArticuloxBodegaNuevo.BODCODIGO = 1;
                    InfoArticuloxBodegaNuevo.PROMAXIMO = 0;
                    InfoArticuloxBodegaNuevo.PROMINIMO = 0;
                    InfoArticuloxBodegaNuevo.PROSTOCK = 0;
                    dc.INV_PRODUCTOXBODEGA.InsertOnSubmit(InfoArticuloxBodegaNuevo);
                }
                dc.SubmitChanges();

                dc.Transaction.Commit();
```

```
                // return intCodigoArticuloRetorno;
            }
            catch (SystemException ex)
            {
                dc.Transaction.Rollback();

LogicaNegocio.Generales.clsGenerales.prcRegistraBitacora("logicaNegocio.Inventarios.clsArticulo",
"prcInsert()", ex, strUserIP);
                throw new ArgumentException("ERROR: Datos no guardados. " + ex.Message.Replace("\"",
"").Replace(Char.ConvertFromUtf32(13), " ").Replace(Char.ConvertFromUtf32(10), ""));
            }
            finally
            {
                dc.Connection.Close();
                dc.Transaction = null;
            }
        }
        //UPDATE PRODUCTO
        public static void prcUpdateProducto(INV_PRODUCTO InfoProducto, INV_PRECIOXPRODUCTO
InfoPrecioxProducto, string[,] arrMaxProducto, string strUserIP)
        {
            sysGestionMedDataContext dc = new Datos.sysGestionMedDataContext();
            try
            {
                dc.Connection.Open();
                dc.Transaction = dc.Connection.BeginTransaction();
                //Envia los datos de usuario e IP a setear en el CONTEXT_INFO para grabar en la
auditoria
                dc.ExecuteCommand(strContexto.Replace("strUserIP", strUserIP));
                //Inserta la informacion del producto
                var obj = dc.INV_PRODUCTO.Single(row =>
row.PROCODIGO.Equals(InfoProducto.PROCODIGO));
                if (InfoProducto.CATCODIGO != null) obj.CATCODIGO = InfoProducto.CATCODIGO;
                if (InfoProducto.PROCODIGOBARRAS != null) obj.PROCODIGOBARRAS =
InfoProducto.PROCODIGOBARRAS;
                if (InfoProducto.PROLOTE != null) obj.PROLOTE = InfoProducto.PROLOTE;
                if (InfoProducto.PRONOMBRECOMERCIAL != null) obj.PRONOMBRECOMERCIAL =
InfoProducto.PRONOMBRECOMERCIAL;
                if (InfoProducto.PROFECHACADUCIDAD != null) obj.PROFECHACADUCIDAD =
InfoProducto.PROFECHACADUCIDAD;
                if (InfoProducto.PROIVA != null) obj.PROIVA = InfoProducto.PROIVA;
                if (InfoProducto.PROINVENTARIO != null) obj.PROINVENTARIO =
InfoProducto.PROINVENTARIO;
                //Guarda el codigo del articulo para el retorno
                Int32 intCodigoArticuloRetorno = Convert.ToInt32(InfoProducto.PROCODIGO);
                //Inserta la informacion del PrecioxProducto
                var InfoPrecioProductoAnterior = dc.INV_PRECIOXPRODUCTO.Single(row =>
row.PROCODIGO.Equals(InfoProducto.PROCODIGO) && row.LISCODIGOESTADOPRECIO.Equals("ACT"));
                InfoPrecioProductoAnterior.LISCODIGOESTADOPRECIO = "INA";
                InfoPrecioProductoAnterior.PXPFECHAVIGENCIAHASTA = System.DateTime.Now;
                InfoPrecioxProducto.PROCODIGO =
Convert.ToInt32(intCodigoArticuloRetorno.ToString());
                InfoPrecioxProducto.PXPCODIGO = dc.INV_PRECIOXPRODUCTO.Any() ?
dc.INV_PRECIOXPRODUCTO.Max(row => row.PXPCODIGO) + 1 : 1;
                InfoPrecioxProducto.PXPFECHAREGISTRO = System.DateTime.Now;
                InfoPrecioxProducto.PXPFECHAVIGENCIADESDE = System.DateTime.Now;
                dc.INV_PRECIOXPRODUCTO.InsertOnSubmit(InfoPrecioxProducto);
                //Inserta la informacion de ARTICULOXBODEGA cuando sea un PRODUCTO
                if (arrMaxProducto != null)
                {
                    for (int i = 0; i <= (arrMaxProducto.Length / 2) - 1; i++)
                    {
                        if (arrMaxProducto[i, 0] != null && arrMaxProducto[i, 1] != null)
                        {
                            if (dc.INV_PRODUCTOXBODEGA.Any(row =>
row.PROCODIGO.Equals(InfoProducto.PROCODIGO) && row.BODCODIGO.Equals(arrMaxProducto[i,
0].ToString())))
                            {
```

```
                                var InfoProductoxBodegaAnterior = dc.INV_PRODUCTOXBODEGA.Single(row
=>     row.PROCODIGO.Equals(InfoProducto.PROCODIGO)     &&     row.BODCODIGO.Equals(arrMaxProducto[i,
0].ToString()));
                                InfoProductoxBodegaAnterior.PROMAXIMO                                =
Convert.ToDecimal(arrMaxProducto[i, 1].ToString());
                            }
                            else
                            {
                                INV_PRODUCTOXBODEGA InfoProductoxBodega = new INV_PRODUCTOXBODEGA();
                                InfoProductoxBodega.PROCODIGO                                        =
Convert.ToInt32(InfoProducto.PROCODIGO);
                                InfoProductoxBodega.BODCODIGO   =   Convert.ToInt32(arrMaxProducto[i,
0].ToString());
                                InfoProductoxBodega.PROMAXIMO  =  Convert.ToDecimal(arrMaxProducto[i,
1].ToString());
                                InfoProductoxBodega.PROMINIMO = 0;
                                InfoProductoxBodega.PROSTOCK = 0;
                                dc.INV_PRODUCTOXBODEGA.InsertOnSubmit(InfoProductoxBodega);
                            }
                        }
                    }
                }
                dc.SubmitChanges();
                dc.Transaction.Commit();
                // return intCodigoArticuloRetorno;
            }
            catch (SystemException ex)
            {
                dc.Transaction.Rollback();

LogicaNegocio.Generales.clsGenerales.prcRegistraBitacora("logicaNegocio.Inventarios.clsArticulo",
"prcInsert()", ex, strUserIP);
                throw new ArgumentException("ERROR: Datos no guardados. " + ex.Message.Replace("\"",
"").Replace(Char.ConvertFromUtf32(13), " ").Replace(Char.ConvertFromUtf32(10), ""));
            }
            finally
            {
                dc.Connection.Close();
                dc.Transaction = null;
            }
        }
    }
}
```

11 CAPA DE PRESENTACIÓN

Ejemplo frmIngresoEgreso.aspx

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using LogicaNegocio;
using System.Data.Linq;
using DevExpress.Web;
using System.Configuration;
using DevExpress.Data.Linq;
```

```
using System.Web.Services;
using System.Web.Script.Services;
using System.Data;

namespace GestionAgenda.Gestion
{
    public partial class frmIngresoEgreso : System.Web.UI.Page
    {
        string strUserIP;
        string strNombreFormulario = "frmIngresoEgreso.aspx";
        string strSinStock;
        DataTable datProductos = new DataTable();
        DataRow dr;
        DataView dv;

        protected void Page_Load(object sender, EventArgs e)
        {
            //Variable usada para enviar el usuario e IP para las AUDITORIAS
            strUserIP      =      GestionAgenda.Global.clsSession.sesUsuario      +      "|"      +
GestionAgenda.Global.clsSession.sesIP;

            if (IsPostBack)
            {
                this.cmbBodega.ReadOnly = true;
                return;
            }

            //En la primera carga, inicializa la tabla de detalle de Factura
            Session["dtIngreso"] = null;


            this.obtenerBodega();
            GestionAgenda.Global.clsGlobal.prcUsuarioLogeado(lblUsuario, lblDepartamento);
        }

        public void obtenerBodega()
        {
            //Lllena el combo de la especialidad
            this.cmbBodega.Items.Clear();
            foreach (var lisHijo in LogicaNegocio.Gestion.clsBodega.prcObtenerBodega())
                this.cmbBodega.Items.Add(lisHijo.BODNOMBRE, lisHijo.BODCODIGO);

        }

        //Limpia todos los controles para iniciar un nuevo ingreso de un Articulo/Servicio
        protected void prcLimpiarDetalleProducto()
        {
            this.txtCodigoProducto.ClientEnabled = false;
            this.cmbProducto.SelectedIndex = -1;
            this.txtCantidadProducto.Text = "";
            this.txtCodigoProducto.Text = "";
            this.txtSaldoActual.Text = "";
            this.cmbMasMenosAjuste.SelectedIndex = 0;
            this.txtPrecioCompra.Text = "";
        }

        protected void prcSumaProductos()
        {
            try
            {
                int intSiExisteRegistro = 0;
                string strArtCodigoIngresa = this.cmbProducto.SelectedItem.Value.ToString();
                string strMasMenos = this.cmbMasMenosAjuste.SelectedItem.Value.ToString();
                string strPrecioCompra = this.txtPrecioCompra.Text;

                decimal decSaldoBodega = 0;
                int j = 0;

                //Si existen registros en la tabla, verifica si ya esta registrado el mismo producto
                if (datProductos.Rows.Count >= 1)
```

```
                {
                    for (j = 0; j < this.datProductos.Rows.Count; j++)
                    {
                        //Si encuentra el producto, suma la cantidad al registro anterior
                        if                      (strArtCodigoIngresa.Trim()                      ==
this.datProductos.Rows[j]["CODIGO"].ToString().Trim()
                        &&                                strMasMenos                            ==
this.datProductos.Rows[j]["MAS_MENOS"].ToString().Trim()
                        &&                             strPrecioCompra                           ==
this.datProductos.Rows[j]["PRECIOCOMPRA"].ToString().Trim()
                        )
                        {
                            this.datProductos.Rows[j]["CANTIDAD"]                                 =
Convert.ToInt16(this.datProductos.Rows[j]["CANTIDAD"])                                            +
Convert.ToInt16(this.txtCantidadProducto.Text.ToString());
                            this.datProductos.Rows[j]["SUBTOTAL"]                                 =
(Convert.ToInt16(this.datProductos.Rows[j]["CANTIDAD"])                                           +
Convert.ToInt16(this.txtCantidadProducto.Text.ToString()))                                        *
Convert.ToInt16(this.datProductos.Rows[j]["PRECIOCOMPRA"]);
                            intSiExisteRegistro = 1;
                            break;
                        }
                    }
                }

                //Si no encontro el producto en la tabla, agrega el registro
                if (intSiExisteRegistro == 0)
                {
                    dr = datProductos.NewRow();
                    dr[0] = Convert.ToInt32(this.txtCodigoProducto.Text);
                    dr[1] = this.cmbProducto.SelectedItem.Text.ToString();
                    dr[2] = this.cmbMasMenosAjuste.SelectedItem.Value.ToString();
                    dr[3] = this.txtCantidadProducto.Text.ToString();
                    dr[4] = this.txtPrecioCompra.Text.ToString();
                    dr[5] = decimal.Round(Convert.ToInt16(this.txtCantidadProducto.Text.ToString())
* Convert.ToDecimal(this.txtPrecioCompra.Text.ToString()), 7);
                    dr[6] = 0;
                    dr[7]    =    this.hidDatosAdicionales["strLlevaIva"].ToString()    ==    "S"    ?
Convert.ToInt16(this.txtCantidadProducto.Text.ToString())                                         *
Convert.ToDecimal(this.txtPrecioCompra.Text.ToString()) : 0;
                    dr[8]    =    this.hidDatosAdicionales["strLlevaIva"].ToString()    ==    "N"    ?
Convert.ToInt16(this.txtCantidadProducto.Text.ToString())                                         *
Convert.ToDecimal(this.txtPrecioCompra.Text.ToString()) : 0;
                    dr[9]    =    this.hidDatosAdicionales["strLlevaIva"].ToString()    ==    "S"    ?
Convert.ToInt16(this.txtCantidadProducto.Text.ToString())                                         *
Convert.ToDecimal(this.txtPrecioCompra.Text.ToString()) * 12 / 100 : 0;
                    datProductos.Rows.Add(dr);
                }
            }
            catch (SystemException err)
            {
                //Obtiene la linea del error
                int intline = LogicaNegocio.Generales.clsGenerales.funObtenerLineaError(err);
                //          GestionAgenda.Global.clsGlobal.prcRegistraBitacora(strNombreFormulario,
"icoAgregar()",    "(Linea:    "    +    Convert.ToString(intline)    +    ")    "    +    err.ToString(),
GestionAgenda.Global.clsSession.sesIP.ToString(),
GestionAgenda.Global.clsSession.sesUsuario.ToString());
                GestionAgenda.Global.clsGlobal.prcEnviaAlerta("Error    en    la    línea:    "    +
Convert.ToString(intline) + ". " + err.Message.ToString());
            }
        }


        protected void btnNuevo_Click(object sender, EventArgs e)
        {
            Response.Redirect(strNombreFormulario);
        }

        protected void btnGrabar_Click(object sender, EventArgs e)
        {
```

```
            try
            {
                if (this.grdPedido.VisibleRowCount == 0)
                {
                    GestionAgenda.Global.clsGlobal.prcEnviaAlerta("ERROR:        Ingrese        las
medicinas/insumos.");
                    return;
                }
                if (this.txtNroDocumentoProveedor.Text != "")
                {
                    var InfoIngresoProductos = LogicaNegocio.Compras.clsCompras.prcTablaCompra();
                    InfoIngresoProductos.COMCODIGO = 0;
                    InfoIngresoProductos.PROVCODIGO                                                =
this.cmbProveedor.SelectedItem.Value.ToString();
                    InfoIngresoProductos.COMFECHAREGRISTRO = System.DateTime.Now;
                    InfoIngresoProductos.COMNROFACTURACOMPRA = this.txtNroDocumentoProveedor.Text;
                    InfoIngresoProductos.TIPDOCCODIGO                                              =
Convert.ToInt16(this.rblTipoDocumento.SelectedItem.Value);

                    //Recorre el grid para obtener los datos del detalle del pedido
                    string[,] arrIngresoDetalles = new string[this.grdPedido.VisibleRowCount, 10];
                    int x = 0;
                    for (int i = 0; i <= this.grdPedido.VisibleRowCount - 1; i++)
                    {
                        arrIngresoDetalles[x,       0]       =       this.grdPedido.GetRowValues(i,
"CODIGO").ToString();//Codigo del Producto
                        arrIngresoDetalles[x,       1]       =       this.grdPedido.GetRowValues(i,
"CANTIDAD").ToString();
                        arrIngresoDetalles[x,       2]       =       this.grdPedido.GetRowValues(i,
"MAS_MENOS").ToString();
                        arrIngresoDetalles[x,       3]       =       this.grdPedido.GetRowValues(i,
"PRECIOCOMPRA").ToString();
                        arrIngresoDetalles[x,       4]       =       this.grdPedido.GetRowValues(i,
"DESCRIPCION").ToString();
                        arrIngresoDetalles[x,       5]       =       this.grdPedido.GetRowValues(i,
"SUBTOTAL").ToString();
                        arrIngresoDetalles[x,       6]       =       this.grdPedido.GetRowValues(i,
"DESCUENTO").ToString();
                        arrIngresoDetalles[x,       7]       =       this.grdPedido.GetRowValues(i,
"TARIFA12").ToString();
                        arrIngresoDetalles[x,       8]       =       this.grdPedido.GetRowValues(i,
"TARIFA0").ToString();
                        arrIngresoDetalles[x, 9] = this.grdPedido.GetRowValues(i, "IVA").ToString();
                        x++;
                    }

                    //Inserta de manera transaccional los registros de UN INGRESO DE INVENTARIO
                    Int16                           intCodigoOrdenEntrega                          =
LogicaNegocio.Compras.clsCompras.prcInsert(InfoIngresoProductos,

arrIngresoDetalles,

this.cmbBodega.SelectedItem.Value.ToString(),

strUserIP);
                    this.lblCodigo.Text = intCodigoOrdenEntrega.ToString();
                    this.grdPedido.SettingsText.Title    =    "DETALLE    DEL    INGRESO    N°    "    +
intCodigoOrdenEntrega.ToString();

                    GestionAgenda.Global.clsGlobal.prcEnviaAlerta("Datos Guardados");
                    this.pnlDatos.Enabled = false;
                    this.pnlArticulos.Enabled = false;
                }
                else
                {
                    GestionAgenda.Global.clsGlobal.prcEnviaAlerta("Nro.  documento  del  Proveedor  en
blanco");
                }
            }
            catch (SystemException err)
```

```
            {
                //Obtiene la linea del error
                int intline = LogicaNegocio.Generales.clsGenerales.funObtenerLineaError(err);
                //          GestionAgenda.Global.clsGlobal.prcRegistraBitacora(strNombreFormulario,
"icoAgregar()",   "(Linea:   "   +   Convert.ToString(intline)   +   ")   "   +   err.ToString(),
GestionAgenda.Global.clsSession.sesIP.ToString(),
GestionAgenda.Global.clsSession.sesUsuario.ToString());
                GestionAgenda.Global.clsGlobal.prcEnviaAlerta("Error   en   la   línea:   "   +
Convert.ToString(intline) + ". " + err.Message.ToString());
            }
        }

        protected void btnModificar_Click(object sender, EventArgs e)
        {

        }

        protected void btnCancelar_Click(object sender, EventArgs e)
        {
            Session["dtIngreso"] = null;
            Response.Redirect(strNombreFormulario);
        }

        protected void btnBuscar_Click(object sender, EventArgs e)
        {

        }

        protected void btnImprimir_Click(object sender, EventArgs e)
        {

        }

        protected void btnEliminar_Click(object sender, EventArgs e)
        {

        }

        protected       void       hidValoresProductoServicios_CustomCallback(object       sender,
CallbackEventArgsBase e)
        {

        }

        protected void cmbProducto_OnSelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                //Si tiene un producto seleccionado
                if (this.cmbProducto.SelectedItem != null)
                {
                    this.txtCodigoProducto.Text = this.cmbProducto.SelectedItem.Value.ToString();
                    var                              lisProductoxBodega                              =
LogicaNegocio.Gestion.cslProducto.prcObtenerSaldoProductoxBodega(Convert.ToInt32(this.txtCodigoProd
ucto.Text), Convert.ToInt16(this.cmbBodega.SelectedItem.Value.ToString()));
                    this.txtSaldoActual.Text  =  lisProductoxBodega.Count  ==  0  ?  "No  Existe"  :
Convert.ToInt16(lisProductoxBodega[0].PROSTOCK).ToString();
                    var                              lisPrecioxProducto                              =
LogicaNegocio.Gestion.cslProducto.prcObtenerPrecioReal(Convert.ToInt32(this.txtCodigoProducto.Text)
);
                    this.txtPrecioCompra.Text  =  lisPrecioxProducto.Count  ==  0  ?  "No  Existe"  :
Convert.ToDecimal(lisPrecioxProducto[0].PXPPRECIOCOMPRA).ToString();
                    this.hidDatosAdicionales["strLlevaIva"]                                          =
LogicaNegocio.Gestion.cslProducto.prcObtenerListProductooxCodigo(Convert.ToInt32(this.cmbProducto.S
electedItem.Value.ToString()))[0].PROIVA.ToString();
                }
                this.txtCantidadProducto.Focus();
            }
            catch (SystemException err)
            {
```

```
                //Obtiene la linea del error
                int intline = LogicaNegocio.Generales.clsGenerales.funObtenerLineaError(err);
                //          GestionAgenda.Global.clsGlobal.prcRegistraBitacora(strNombreFormulario,
"icoAgregar()",   "(Linea:   "   +   Convert.ToString(intline)   +   ")   "   +   err.ToString(),
GestionAgenda.Global.clsSession.sesIP.ToString(),
GestionAgenda.Global.clsSession.sesUsuario.ToString());
                GestionAgenda.Global.clsGlobal.prcEnviaAlerta("Error   en   la   línea:   "   +
Convert.ToString(intline) + ". " + err.Message.ToString());
            }
        }

        protected void icoAgregar_Click(object sender, EventArgs e)
        {
            try
            {
                this.grdPedido.Visible = true;

                //Verificar en session si esta la tabla
                if (Session["dtIngreso"] != null)
                {
                    //this.dt.Rows.Add(new Object[] { });
                    datProductos = (DataTable)Session["dtIngreso"];
                }
                else
                {
                    //Activa funciones de grabado y crea las columnas de la tabla que estara en
session
                    datProductos.Columns.Add(new DataColumn("CODIGO", typeof(int)));
                    datProductos.Columns.Add(new DataColumn("DESCRIPCION", typeof(string)));
                    datProductos.Columns.Add(new DataColumn("MAS_MENOS", typeof(string)));
                    datProductos.Columns.Add(new DataColumn("CANTIDAD", typeof(string)));
                    datProductos.Columns.Add(new DataColumn("PRECIOCOMPRA", typeof(string)));
                    datProductos.Columns.Add(new DataColumn("SUBTOTAL", typeof(string)));
                    datProductos.Columns.Add(new DataColumn("DESCUENTO", typeof(string)));
                    datProductos.Columns.Add(new DataColumn("TARIFA12", typeof(string)));
                    datProductos.Columns.Add(new DataColumn("TARIFA0", typeof(string)));
                    datProductos.Columns.Add(new DataColumn("IVA", typeof(string)));
                }

                this.grdPedido.Visible = true;
                //Suma productos o agrega el producto a la tabla
                this.prcSumaProductos();
                datProductos.AcceptChanges();
                dv = new DataView(datProductos);
                this.grdPedido.DataSource = dv;
                this.grdPedido.DataBind();
                Session["dtIngreso"] = datProductos;
                this.prcLimpiarDetalleProducto();

                this.cmbProducto.Focus();
            }
            catch (SystemException err)
            {
                //Obtiene la linea del error
                int intline = LogicaNegocio.Generales.clsGenerales.funObtenerLineaError(err);
                //          GestionAgenda.Global.clsGlobal.prcRegistraBitacora(strNombreFormulario,
"icoAgregar()",   "(Linea:   "   +   Convert.ToString(intline)   +   ")   "   +   err.ToString(),
GestionAgenda.Global.clsSession.sesIP.ToString(),
GestionAgenda.Global.clsSession.sesUsuario.ToString());
                GestionAgenda.Global.clsGlobal.prcEnviaAlerta("Error   en   la   línea:   "   +
Convert.ToString(intline) + ". " + err.Message.ToString());
            }
        }

        protected void icoCancelar_Click(object sender, EventArgs e)
        {
            try
            {
                this.prcLimpiarDetalleProducto();
                this.cmbProducto.Focus();
```

```
            }
            catch (SystemException err)
            {
                //Obtiene la linea del error
                int intline = LogicaNegocio.Generales.clsGenerales.funObtenerLineaError(err);
                //          GestionAgenda.Global.clsGlobal.prcRegistraBitacora(strNombreFormulario,
"icoAgregar()",   "(Linea:   "   +   Convert.ToString(intline)   +   ")   "   +   err.ToString(),
GestionAgenda.Global.clsSession.sesIP.ToString(),
GestionAgenda.Global.clsSession.sesUsuario.ToString());
                GestionAgenda.Global.clsGlobal.prcEnviaAlerta("Error   en   la   línea:   "   +
Convert.ToString(intline) + ". " + err.Message.ToString());
            }
        }

        protected             void             grdPedido_OnRowDeleting(object             sender,
DevExpress.Web.Data.ASPxDataDeletingEventArgs e)
        {
            try
            {
                int intIndice = 0;
                datProductos = (DataTable)Session["dtIngreso"];
                foreach (DataRow dr in this.datProductos.Rows)
                {
                    //Para usar esta propiedad en el Gridview debe configurarse:
                    // <SettingsBehavior AllowFocusedRow="True" AllowSelectByRowClick="False" /> y el
KeyFieldName debe estar establecido tambien
                    if (intIndice == (sender as ASPxGridView).FocusedRowIndex)
                    {
                        dr.Delete();
                        datProductos.AcceptChanges();
                        dv = new DataView(datProductos);
                        this.grdPedido.DataSource = dv;
                        this.grdPedido.DataBind();
                        Session["dtIngreso"] = datProductos;
                        break;
                    }
                    else
                        intIndice++;
                }
                this.prcLimpiarDetalleProducto();
            }
            catch (SystemException err)
            {
                GestionAgenda.Global.clsGlobal.prcEnviaAlerta(err.Message.ToString());
            }
            finally
            {
                e.Cancel = true;
            }
        }

        protected          void          grdPedido_OnCustomSummaryCalculate(object          sender,
DevExpress.Data.CustomSummaryEventArgs e)
        {
            ASPxSummaryItem decSubtotal = (sender as ASPxGridView).TotalSummary["SUBTOTAL"];
            ASPxSummaryItem decDescuento = (sender as ASPxGridView).TotalSummary["DESCUENTO"];
            ASPxSummaryItem decIva = (sender as ASPxGridView).TotalSummary["IVA"];

            Decimal                                  subtotal                                  =
Convert.ToDecimal(((ASPxGridView)sender).GetTotalSummaryValue(decSubtotal));
            Decimal                                 descuento                                  =
Convert.ToDecimal(((ASPxGridView)sender).GetTotalSummaryValue(decDescuento));
            Decimal iva = Convert.ToDecimal(((ASPxGridView)sender).GetTotalSummaryValue(decIva));

            e.TotalValue = subtotal - descuento + iva;

        }

        protected void linqProducto_OnSelecting(object sender, LinqDataSourceSelectEventArgs e)
        {
```

```
            //Limpia el combo
            if (this.cmbProducto.Items.Count >= 1)
            {
                this.cmbProducto.Items.RemoveAt(0);
                this.cmbProducto.SelectedIndex = -1;
            }
            //Obtiene los PRODUCTOS/SERVICIOS
            e.Result =  LogicaNegocio.Gestion.cslProducto.prcObtenerProductosxCategorias();
            this.cmbProducto.TextField = "PRONOMBRECOMERCIAL";
            this.cmbProducto.ValueField = "PROCODIGO";

        }
    }
}
```