UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 7

TÍTULO: Implementación del diseño de un Sistema de Información Farmacéutica

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

AUTOR: Yojanier Carvajal Pérez

TUTOR: Ing. Arianne Méndez Mederos

La Habana, Julio 2007

"Año 49 del triunfo de la Revolución."

Frase

En los momentos de crisis, sólo la imaginación
es más importante que el conocimiento.

Albert Einstein.

Declaración de Autoría

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los 6 días del mes de Julio del año 2007.

Yojanier Carvajal Pérez **Ing. Arianne Méndez Mederos**

_____ _____

Firma del Autor Firma del Tutor

Datos de contacto.

Tutor

Ing. Arianne Méndez Mederos, recién graduada de la Universidad de las Ciencias Informáticas y actual profesora de la facultad 7, natural de Villa Clara, e-mail arianne@uci.cu

Agradecimientos

A mi madre, por ser el apoyo que siempre necesité para llegar a este gran momento.

A mi padre.

A mi hermano.

A mi familia en general, porque siempre se preocuparon por mí.

A María de Lourdes, por estar siempre conmigo.

A la Ing. Gema Feo Gamio, por la ayuda incondicional que me dió.

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

A todos los que de una forma u otra estuvieron presentes en cada paso que di para llegar a este momento.

Dedicatoria

Dedico este trabajo a mi madre, por ser quien compartió conmigo este sueño, me hizo entender que por lejos que estuviera la meta o por difícil que fuera el camino a recorrer, nunca debía abandonarlo; me enseñó que si las cosas me salen mal una vez no debo dejarlo, sino volverlo a hacer. Me dijo, que sin sacrificio no hay un logro digno; por ser quien es, le dedico este trabajo.

Yojanier.

Resumen

El presente trabajo tiene como objetivo implementar una aplicación Web para automatizar los principales procesos asociados con la prestación de servicios de las farmacias hospitalarias del país. Actualmente, el proceso de gestión de información en las farmacias hospitalarias se hace de forma manual, lo que trae consigo: un manejo no confiable de los procesos, un mayor consumo de recursos para contabilizar los datos y se alarga el período de actualización de la información.

Para la implementación del sistema se utiliza la plataforma .NET y por ende el lenguaje de desarrollo del lado del servidor ASP.NET, como lenguaje de desarrollo del lado del cliente se usará JavaScript. También se hace uso de la tecnología AJAX, se utilizará la herramienta de desarrollo Visual Studio 2005 y C# como lenguaje base de programación. La aplicación será compilada sobre framework .NET 2.0.

Con el desarrollo de este sistema se espera que los involucrados en los procesos que se llevan a cabo en las farmacias hospitalarias cuenten con una herramienta de trabajo capaz de gestionar de forma eficiente la información manejada. La aplicación debe facilitar el uso controlado, organizado y racional a los productos farmacéuticos. El sistema posibilita que el flujo de información, que sea más confiable y fácil de manipular.

Palabras claves:
Implementación, Farmacia.

Índice

Índice.

Introducción

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) están cada vez más presentes en el sector de la sanidad. La necesidad de mejorar los servicios de planificación sanitaria y los recursos hospitalarios han hecho necesaria la integración de las nuevas tecnologías en el sector de salud.

Con el uso de las nuevas tecnologías surgen los Sistemas de Información Hospitalaria (SIH) que son sistemas encargados de la gestión de la información en las áreas de salud en las cuales el conjunto de información es muy grande y mediante un sistema automatizado el flujo se hace más rápido y organizado.

“Con el análisis, diseño y desarrollo de módulos específicos de un sistema hospitalario en 1988, se inició en la Universidad Autónoma de Guadalajara (UAG), el desarrollo de sistemas informáticos para hospitales, estos módulos fueron: Sistema de control de cargos para pacientes hospitalizados, el cual tenía como objetivo el control de datos administrativos y de consumos del paciente en su estancia en el hospital”. [1]

“Por otro lado y sin integridad con el anterior proyecto, se inició el Sistema de control del expediente clínico, el mismo pretendía controlar los datos del paciente desde el punto de vista clínico”. [2]

“Estas soluciones que en su momento se consideraron óptimas para cubrir las necesidades básicas para la gestión de datos dentro del hospital, mostraron que en verdad no eran integrales y no eran totalmente efectivas para gestionar todo el cúmulo de datos que se mueven dentro de un hospital”. [3]

En la década del 90, se trazaron nuevas estrategias dentro del sistema de salud cubano en la cual el objetivo principal era la informatización de la salud cubana y por ende el desarrollo de herramientas para lograr este fin, se propone la integración de un SIH, para el desarrollo de la salud cubana.

Paso a paso se fueron uniendo esfuerzos entre diferentes centros hasta crear el “SIH”, el cual ha sido objeto de mantenimientos para mejorar sus prestaciones dentro de la Universidad de las Ciencias Informáticas (UCI).

Una de las principales prioridades del estado cubano es la salud de la población, por lo que se dedican gran cantidad de recursos económicos para fomentar avances en ese sentido. Cada año se crean o se

restauran diversos centros de salud, que van desde un policlínico, un consultorio médico de la familia hasta un hospital especializado, la compra de tecnología de punta para mejorar la asistencia médicas en los centros del país es otra de las prioridades principales del estado cubano.

Otra visión de lo antes mencionado es el trabajo de las escuelas de “Ciencias Medicas” en las cuales cada año se gradúan cientos de médicos que hacen que la salud llegue a todos los rincones del país.

Un sistema que agilice el trabajo para la gestión de la información en la farmacia tiene gran importancia debido a que en Cuba la salud juega un papel primordial y por consiguiente la farmacia viene a tomar un papel protagónico debido a que es la encargada de distribuir los medicamentos para el buen funcionamiento del hospital.

En los hospitales uno de los grandes problemas que existe es el uso excesivo del papel para realizar las operaciones internas y externas en las farmacias, y esto esta dado porque para tener un buen control de los medicamentos que entran o salen de la farmacia, deben estar muy bien registrado para que no ocurran pérdidas, mal uso o vencimiento de los productos; que de ocurrir pueden afectar seriamente el buen funcionamiento de la institución.

La Farmacia Hospitalaria tiene a su cargo las entregas de medicamentos a otras unidades internas dentro del hospital que son las encargadas de realizar los pedidos de medicamentos a la farmacia en relación con los pacientes que tienen internados, así como la recepción de algunos medicamentos que se devuelven por parte de las salas.

La farmacia se encarga también de la entrega de medicamentos a pacientes que no están ingresados en el hospital pero que han sido tratados en el mismo a los que al presentar una indicación del medico por el cual fue atendido se le hace entrega del medicamento. La farmacia es la encargada de gestionar los pedidos al almacén o a empresas dedicadas a la producción de medicamentos, otra función de la institución es la creación de medicamentos para ser entregados a los pacientes que lo necesiten.

Se realiza un resumen de las operaciones que se llevan a cabo en las diferentes áreas del hospital a las cuales se les entrega algún producto, con el objetivo de contabilizar el uso de los medicamentos por cada paciente ingresado en el hospital.

Contabiliza y supervisa todos los movimientos de medicamentos que se realizan en la farmacia a través de los controles de stock que se realizan a cada medicamento, según sus parámetros de consumos de los

mismos, cada cierto periodo de tiempo. Además de inventariar y supervisar la existencia de los medicamentos en la farmacia comparando las entradas y salidas de los medicamentos.

Todo este proceso es algo engorroso de seguir debido que el trabajo con un gran número de papeles puede facilitar errores o perdidas en los documento y por ende una perdida irrecuperable de la información además de que los médicos pueden recetar algún medicamento no disponible en la farmacia debido a que el funcionario posea una mala información sobre los recursos que tenga la farmacia.

En la actualidad los sistemas de información se han convertido en una herramienta fundamental para el trabajo con gran flujo de datos que para trabajarlos de la forma convencional afectaría mucho el proceso ya que se volvería engorroso debido a que se demoraría mucho, además aumentarían las probabilidades de que la información se cambie por error o se pierda, por lo que sería necesario para agilizar el trabajo un sistema que pudiera brindar un servicio para mejorar este problema.

Un sistema de información para la gestión integral de la farmacia hospitalaria es de gran importancia ya que vendría a resolver los problemas de papeleo y de trabajo excesivo que tienen como principal problema las farmacias de los hospitales cubanos.

Problema de investigación:

"¿Cómo resolver desde la perspectiva informática todos los aspectos de la gestión de la farmacia hospitalaria?".

Objeto de estudio:

Los procesos de gestión farmacéutica que se desarrollan en el país.

Campo de acción:

Todos los aspectos de la farmacia hospitalaria donde se recogen los procesos para la gestión de los medicamentos.

Objetivo general:

Implementar un sistema capaz de gestionar todas las acciones requeridas en la farmacia de un hospital.

Introducción.

Se han planteado las siguientes **tareas de investigación** para el trabajo:

- Estudiar otros sistemas informáticos para la gestión de farmacia existentes.
- Investigar tendencias actuales de la tecnología en el desarrollo de *software*.
- Estudiar lenguaje de desarrollo a utilizar.
- Estudiar los estándares de codificación y de comunicación establecidos.
- Estudiar el lenguaje del modelado para una correcta interpretación de las clases del diseño.
- Realizar la implementación utilizando los patrones de diseño establecidos en el análisis.

El presente documento quedara estructurado por tres capítulos de la siguiente manera:

Capítulo 1 Fundamentación teórica. Se analizan otros sistemas en uso a nivel nacional e internacional, el estado del arte en las tendencias sobre la implementación de sistemas para resolver el problema planteado, así como una explicación de los requisitos que debe cumplir la solución propuesta.

Capítulo 2 Descripción y análisis de la solución propuesta. Se hará una valoración crítica del diseño propuesto por el analista, un análisis sobre componentes, módulos o implementaciones que podrían ser reutilizadas para dar solución al problema propuesto, un análisis del estándar de codificación a utilizar así como la descripción de las clases implementadas para dar solución al problema.

Capítulo 3 Validación de la solución propuesta. Se realizará un análisis y descripción de los posibles “Test de unidades” para validar la solución propuesta. También, una descripción de los valores utilizados para los test, así como una evaluación de la ejecución y los resultados obtenidos.

Cada capítulo consta de una breve introducción donde se explica el tema a desarrollar en el capítulo y las conclusiones donde se explicarán los resultados obtenidos.

Capítulo 1

1.1-Introducción

En este capítulo se abordarán otros sistemas implementados para ser usados en las farmacias hospitalarias a nivel internacional y dentro del país, también, las tendencias mundiales para programar estos sistemas, los lenguajes y plataformas más usados a nivel mundial así como los usados en la universidad así como las técnicas y arquitecturas que serán usadas para dar solución al problema que concierne en el tema tratado. Además se menciona las tecnologías elegidas para dar solución al problema.

1.2-Descripción del sistema

Debido a los problemas generados a partir del manejo de la información en las farmacias hospitalarias que pueden traer consigo pérdidas o mala gestión de los recursos médicos de los cuales la institución es responsable se hace necesario la implementación de un Sistema de Gestión Farmacéutica que automatice todos los procesos que se llevan a cabo como parte del trabajo cotidiano dentro de un centro farmacéutico en el hospital.

El sistema debe cumplir con un manejo óptimo y confiable de las informaciones y procesos dentro de la farmacia, una disminución de los recursos que se utilizan para contabilizar el trabajo farmacéutico, así como un considerable ahorro del tiempo en que se realizan todas las actividades de la farmacia. El sistema debe posibilitar un mayor flujo de información, mucho más confiables y fáciles de manipular que la existente, mayor variedad y confiabilidad de los cálculos de consumo de los medicamentos.

Posibilitará una dispensación mucho más rápida y ágil de los medicamentos, los cuales se podrían realizar directamente desde las salas. Las gestiones de los pedidos de la farmacia a las unidades proveedoras estarían dadas por los cálculos hechos por el sistema sobre los indicadores de consumo de los productos cuyas referencias estarían guardadas en la base de datos de la aplicación.

Se dispondría de un mayor flujo de información de todas las acciones que se realizan en la farmacia con todos los reportes disponibles de todos los flujos de trabajo. El sistema tendría un mayor flujo de información mucho mas organizada de las dispensaciones que se generan hacia cada paciente que se encuentra hospitalizado en el hospital, así como de aquellos pacientes que no se encuentran hospitalizados,

El sistema se integraría al sistema de gestión hospitalaria que opera en el hospital, posibilitando una comunicación mucho más rápida y precisa entre las diferentes áreas hospitalarias del centro.

1.3-Sistemas en uso en la actualidad

Actualmente muchos hospitales del mundo utilizan diferentes aplicaciones (*software*) para gestionar el gran cúmulo de información que en ellos se mueve, un sector hospitalario donde se puede decir que es uno de los principales lugares donde se gestiona información es la farmacia, por esto desde hace tiempo se viene usando a nivel mundial diferentes *software* para hacer la gestión de la información.

1.3.1-Sistemas en uso a nivel internacional.

En la actualidad existen en funcionamiento múltiples sistemas para la gestión farmacéutica entre los que se destacan: BIOCOM, Etron Farmacia, GESTIFARMA, FARHOS entre otros.

BIOCOM: es un sistema de uso general en el hospital que cuenta con varios módulos, entre los cuales existe uno para uso farmacéutico llamado “Módulo de Farmacia” en el cual se gestiona casi todo el cúmulo de información farmacéutica. Este sistema esta implementado en Basic, además de que maneja ICD10, LOINC, DICOM, ATM, HL7 como estándares de salud utilizados. [4]

Etron Farmacia: este sistema en particular es compatible con cualquier sistema operativo de la familia Windows, tiene una gran escalabilidad ya que se adapta a los requerimientos de los clientes, está implementado por Borland Database Engine compatible con SQL92. Estos sistemas están implementados para la farmacia en general. [5]

GESTIFARMA: al igual que los otros productos ya mencionados es un *software* propietario que incluye módulos como la Gestión Integral que controla la atención farmacéutica, las existencias, las ventas, las

compras, también presenta las gestiones de stock entre otras. Este *software* presenta similitud con el trabajo farmacéutico cubano, pero se diferencia en la forma del tratamiento de las operaciones, este sistema está implementado para el uso sobre sistema operativo Windows. [6]

FARHOS: incluye gestión de compras y almacén, control de adquisiciones entre otras tareas. Todos estos sistemas están creados bajo la política de *software* propietario. Este sistema esta implementado bajo arquitectura cliente-servidor, es compatible la parte del cliente con cualquier sistema operativo de la familia Windows, desde Windows 95 en adelante.

El servidor puede tener instalado desde Windows 95 hasta Unix. Como sistema gestor de base de datos se ha seleccionado Interbase 6, y como herramientas de desarrollo cliente MS-Access 2000, aunque se puede instalar sobre Oracle 8 en sistemas muy grandes. [7]

CARE2X: es un sistema de uso general que posee un módulo de farmacia, este sistema pertenece a la familia de *software* libre, es por este hecho que el sistema que fue creado en Alemania, halla sido reestructurado en Argentina.

La primera versión de este *software* fue implementada bajo PHP4, con servidor de base de datos MySQL, corría sobre servidor Web Apache 1.X o 2.X, pero ya se pueden encontrar versiones de este *software* que se encuentran implementado en PHP5 y PostgreSQL. Este sistema fue usado en Cuba cuando la Misión Milagro, teniendo buenos resultado en el trabajo sobre todo aquí en la UCI que se usó el módulo para inscribir pacientes. [8]

1.3.2-Sistemas en uso a nivel nacional

En Cuba entre los sistemas más usados se encuentran el Galen, el SIH

Galen: que cuenta con un módulo de farmacia, el cual está implementado en ambiente Delphi, cuyo lenguaje base es Pascal, esta herramienta es poco usada en la actualidad. Este sistema ha sido objeto de mantenimiento dentro de la UCI, este *software* ha sido usado en algunos hospitales cubanos.

Este *software* desarrollado por Softel se encarga sobre todo de las gestiones del almacén para la distribución de medicamentos en el hospital, pero no cuenta con la dispensación de medicamentos a los

pacientes externos, el archivo de la información referente a la entrega de cada medicamento, a cada paciente ya sean internos y externos. Por lo que sería necesario implementar un módulo que cumpla con estas actividades que son tan importantes para el buen funcionamiento de la farmacia hospitalaria.

SIH: Se encuentra implementado en Visual Basic y Delphi y como gestor de base de datos el SQL Server 2000. Este sistema ha sido objeto de varios mantenimientos dentro de la UCI por un grupo de desarrollo de la empresa Softel en conjunto con un grupo de estudiantes del centro de altos estudios.

La mayoría sistemas tratados en este epígrafe pertenecen al grupo de *software* propietario, esto es una de las principales desventajas que tienen para ser usados en Cuba, la solución propuesta plantea la implementación de una nueva aplicación, que aunque cuenta con características similares a algunos de los ejemplos mostrados, para su desarrollo no será posible reutilizar el código de estas aplicaciones debido a que están implementados cada uno en lenguajes muy diferentes.

1.4-Tecnologías a usar

Para el desarrollo de un *software* es necesario hacer un estudio de tecnologías que puedan ser usadas para dar solución al problema en cuestión.

1.4.1-Arquitectura Cliente – Servidor

Esta arquitectura consiste básicamente en las peticiones que se realizan por parte de una aplicación (cliente) a otra aplicación (Servidor), esta técnica se puede aplicar en la misma computadora pero resulta más ventajosa cuando se usa en un sistema de varios usuarios distribuidos en una red de computadoras, al estar centralizada la gestión de la información y estar separada las responsabilidades, se facilita el diseño del sistema.

El servidor no tiene que ser ejecutado necesariamente en una maquina, ni necesariamente tiene que ser una sola aplicación, esto esta dado a que la separación entre el cliente y el servidor es de tipo lógica. Algunas ventajas de esta arquitectura son: los accesos, recursos y la integridad de los datos, que son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no puede dañar el sistema.

Se puede aumentar la capacidad de clientes y de servidores por separado, el tráfico en la red se reduce grandemente debido a que el cliente se comunica con el servidor por protocolo de alto nivel de abstracción como es el caso de SQL. [9]

1.4.2-Aplicación Web

Con el surgimiento de Internet surgieron grandes cambios en cuanto al uso y acceso a la información desde cualquier parte del mundo, cada vez es necesario aplicaciones más rápidas, ligeras y robustas que puedan ser usadas por cualquier usuario sin importar la hora o lugar. [10]

Estas aplicaciones están basadas principalmente en la arquitectura cliente servidor. Las aplicaciones Web presentan grandes ventajas en su empleo, ejemplo de esto es:

- Para su uso basta con su instalación en un servidor, ya que el cliente puede acceder a ella mediante un navegador.
- Alta disponibilidad, ya que pueden ser usadas por el usuario en cualquier parte del mundo donde tenga acceso a Internet y a cualquier hora.
- Se reduce el costo de mantenimiento debido a que el mantenimiento sólo debe realizarse en el servidor.
- Son compatibles con cualquier sistema operativo. [11]

1.4.3-Lenguajes y plataformas de desarrollo

En la actualidad existen varios lenguajes vinculados al desarrollo de una aplicación Web, los cuales se pueden definir en dos grupos: lenguajes de desarrollo del lado del servidor y lenguaje de desarrollo del lado del cliente.

Lenguajes de desarrollo del lado del servidor: Estos son lenguajes que se ejecutan del lado del servidor como su nombre lo indica, se basa en códigos encargados de construir páginas que serán enviadas como respuesta al cliente, además de ser los encargados de acceder a la base de datos o a otros recursos de la red, estos códigos serán invisibles a los usuarios ya que al ser solicitados a través del servidor Web se generan en código HTML que son entendibles y mostrados por un navegador.

Algunos de estos lenguajes:

PHP: es un lenguaje de programación usado generalmente para la creación de contenido para sitios Web. PHP es un acrónimo recurrente que significa "Hypertext Pre-processor" y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web. [12]

PHP presenta múltiples ventajas para el programador ya que es posible conectarse con la mayoría de los gestores de base de datos sobre todo con MySQL, puede leer y manipular datos provenientes de formularios HTML y de otras fuentes.

Es muy fácil encontrar documentación sobre PHP sobre todo en Internet debido a su popular uso en estos tiempos, este lenguaje pertenece a la familia del *software* libre y es por esto que su uso actualmente ha alcanzado un gran favoritismo.

En este lenguaje no es necesario tener que definir las variables que se van a usar, esto brinda mayor rapidez a la hora de la codificación pero puede ser un desventaja debido a que con facilidad pueden ser introducidos errores al código por una mala asignación de las variables.

Sin embargo al realizar todo el trabajo en el lado del servidor y no delegar nada en el cliente, el trabajo se vuelve ineficiente a medida que aumenten las peticiones de información, al mezclar código HTML con código PHP se puede volver un poco ilegible la codificación, a la hora de darle mantenimiento si no se tiene un buen diseño, el trabajo se vuelve engorroso. [13]

PERL: (Practical Extraction and Report Language)Es un lenguaje script, por lo que no hace falta compilar el programa escrito ya que casi siempre se compila parcialmente al comienzo de su ejecución, este lenguaje es muy usado para la creación de pequeños programas que sirven de filtros para obtener información en ficheros y realizar búsquedas, las aplicaciones se desarrollan bastante rápido, además de estar disponible en muchas plataformas y sistemas operativos y de pertenecer también a la familia de *software* libre por lo que su código fuente está disponible para cualquier usuario.

No obstante PERL no es todo lo perfecto que se cree ya que utiliza muchos recursos de la computadora, cualquiera puede ver el código fuente de la aplicación ya que es un lenguaje interpretado y no compilado, este lenguaje también le ofrece muchas libertades al programador por lo que la codificación puede hacerse un poco ilegible para un posterior mantenimiento. [14]

ASP: A pesar de ser un *software* propietario que puede ser una de sus principales desventajas debido a que pertenece a la Microsoft, por lo que está creado para correr en sistema operativo Windows e Internet Information Server (IIS).

ASP ofrece grandes posibilidades para la creación de aplicaciones Web, ya que una página ASP es básicamente una página HTML con código incrustado que será ejecutada del lado del servidor por lo tanto siempre que se solicite una información determinada, esta se ejecutará del lado del servidor y entonces la respuesta será entregada a la parte del cliente, por lo que dentro de la página también se puede usar código XML. [15]

Esto muestra que usando ASP el cliente no tiene acceso nunca al código que se ejecuta y así se evitan intrusiones no deseadas y se aumenta la seguridad del sistema, las llamadas a la base de datos se hacen a través de objetos por lo que el proceso resulta más rápido, las conexiones se puede integrar con bases de datos postgreSQL, MySQL, SQL Server así como acces2000, se puede programar en visual Basic u otro lenguaje para la programación de aplicaciones Web.

De los lenguajes de desarrollo del lado del servidor expuestos anteriormente, los que más se usan son PHP y ASP el primero por ser un *software* libre además de sus facilidades de desarrollo, y el segundo por su rapidez con que se pueden desarrollar aplicaciones de este tipo.

Lenguajes de desarrollo del lado del cliente: Los lenguajes del lado del Cliente por otra parte, como su nombre lo indica son los que se ejecutan en el cliente o navegador. Estos son los encargados de darle dinamismo a la página sin necesidad de enviar la información al servidor para realizar las operaciones que se requieran.

Estos lenguajes son interpretados, pueden acceder a la información HTML que se muestra en un navegador pudiendo modificarla o actualizarla según las necesidades de los programadores.

Algunos ejemplos de lenguajes de desarrollo del lado del cliente:

JavaScript: Lenguaje de programación del lado del cliente más utilizado por ser compatible con la mayoría de los navegadores modernos. Es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Además, se puede acceder a los elementos que

componen la página Web, permitiéndole al programador modificar el contenido de la página dinámicamente. [16]

VisualBasicScript: Este lenguaje a diferencia de JavaScript es compatible solamente con el Internet Explorer, debido que es desarrollado por Microsoft. Está basado en Visual Basic. Sin embargo posee toda la funcionalidad que brinda JavaScript. [17]

De los lenguajes de desarrollo del lado del cliente anteriormente expuesto el más usado en la UCI es JavaScript debido a su compatibilidad con casi todos los navegadores.

1.4.4-Servidores Web

Internet Information Server (IIS):

Es el principal servidor de aplicaciones Web de Microsoft. Sus principales funcionalidades son la publicación de sitios y aplicaciones Web, sitios FTP (File Transfer Protocol), SMTP (Simple Mail Transport Protocol) y Servicios de noticias. Dispone de soporte necesario para crear páginas en ASP.

Su principal problema está en su pobre seguridad, el cual, fue resuelto posteriormente en la versión 6.0, donde Microsoft ha cambiado el comportamiento de los controles ISAPI preinstalados. Es un sistema esencial destinado a la utilización de los servicios de Internet basado en la plataforma Windows. [18]

Apache:

Es el principal servidor Web del mundo del *software* libre. Después de la segunda mitad de la década del 90 ha tomado un gran auge en las aplicaciones Web, fundamentalmente en el soporte de aplicaciones programadas en PHP. Entre sus principales características están la flexibilidad para configurar los mensajes de error, contiene sus propias bases de datos de autenticación y negociado de contenido. [19]

1.4.5-Ajax (Asynchronous JavaScript + XML)

AJAX son las siglas de Asynchronous JavaScript And XML (JavaScript y XML asíncronos, donde XML es un acrónimo de eXtensible Markup Language), es una tecnología de desarrollo Web para crear aplicaciones interactivas sin la necesidad de cargar la página completamente entre una solicitud y otra.

Estas se ejecutan en el navegador, y mantiene comunicación asíncrona con el servidor en segundo plano mediante un canal de conexión.

Debido a que la página Web no se refresca completamente, si no solamente se acceden a elementos de la misma mediante JavaScript u otro lenguaje del lado del cliente, se gana en rapidez e interactividad. [20]

1.4.6-Plataforma .NET

Para crear páginas ASP una de las plataformas más usadas es la plataforma .NET, esta es una plataforma creada por Microsoft y orientada a la creación de *software* para Internet, el corazón de esta plataforma es el CLR (Common Language Runtime).

Esto es muy similar a una máquina virtual que gestiona la ejecución de las aplicaciones escritas, esto utiliza varios servicios para facilitar el desarrollo, la seguridad y la confiabilidad de las aplicaciones, entre estos servicios destacan: un modelo de programación sencillo completamente orientado a objetos, ejecuciones multiplataformas y multilenguajes, etc. [20]

ASP. NET: es un conjunto de tecnologías de desarrollo de aplicaciones Web comercializado por Microsoft. Es usado por programadores para construir sitios Web domésticos, aplicaciones Web y servicios XML. Forma parte de la plataforma .NET de Microsoft y es la tecnología sucesora de la tecnología Active Server Pages (ASP).

ASP.NET contiene algunas mejoras en cuanto a posibilidades del lenguaje y a la rapidez con que funcionan. Además constituye una de las formas más rápidas de crear aplicaciones Web además de presentar una gran escalabilidad, es por esto que los desarrolladores lo tengan como una de las primeras opciones a la hora de desarrollar aplicaciones Web. [21]

Algunas ventajas de ASP.NET:

- Mejor rendimiento.
- Compatibilidad con herramientas de primer nivel.
- Eficacia y flexibilidad.
- Simplicidad.

- Facilidad de uso.
- Escalabilidad y disponibilidad.
- Posibilidad de personalización y extensibilidad.
- Seguridad.

Algunas desventajas de ASP.NET:

- Para que todo ocurra en una página Web, es habitual escribir una gran cantidad de código para resolver necesidades sencillas.
- ASP.NET impone un cierto orden sobre el modelo de programación estándar ASP.
- ASP.NET separa la porción basada en script de una página Web de su contenido.
- La tercera limitación en el desarrollo con ASP es que con el tradicional, se utilizan lenguajes de scripting no tipados como VBScript o JScript.

Dentro de los posibles lenguajes con los que se desarrollan aplicaciones dentro de esta plataforma está el C#, este lenguaje fue creado específicamente para ser usado dentro de .NET.

El lenguaje C# combina elementos de varios lenguajes entre los que se encuentran: C++, Java, Object Pacal, Visual Basic. Aunque C# forma parte de la plataforma .NET, ésta es una interfaz de programación de aplicaciones; mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma.

Este lenguaje es orientado a objetos por lo que incluye herencia, polimorfismo, encapsulación, en esta ultima propiedad además de usar los mismos modificadores que otros lenguajes como son el “private”,”public”,”protected”, incluye un nuevo modificador llamado “internal”, que puede combinarse con protected e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.

C# puedes ser también orientado a componentes ya que incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas,

es decir que este lenguaje permite definir propiedades, eventos o atributos mas cómodamente que en otros lenguajes. [22]

1.4.7- Biblioteca de clases de .NET

Cuando se está programando una aplicación muchas veces se necesitan realizar acciones como manipulación de archivos, acceso a datos, conocer el estado del sistema, implementar seguridad, etc. El Framework organiza toda la funcionalidad del sistema operativo en un espacio de nombres jerárquico de forma que a la hora de programar resulta bastante sencillo encontrar lo que se necesita.

El Framework .NET posee un sistema de tipos universal, denominado Common Type System (CTS). Este sistema permite que el programador pueda interactuar los tipos que se incluyen en el propio Framework (biblioteca de clases de .Net) con los creados por él mismo (clases).

De esta forma se aprovechan las ventajas propias de la programación orientada a objetos, como la herencia de clases predefinidas para crear nuevas clases, o el polimorfismo de clases para modificar o ampliar funcionalidades de clases ya existentes.

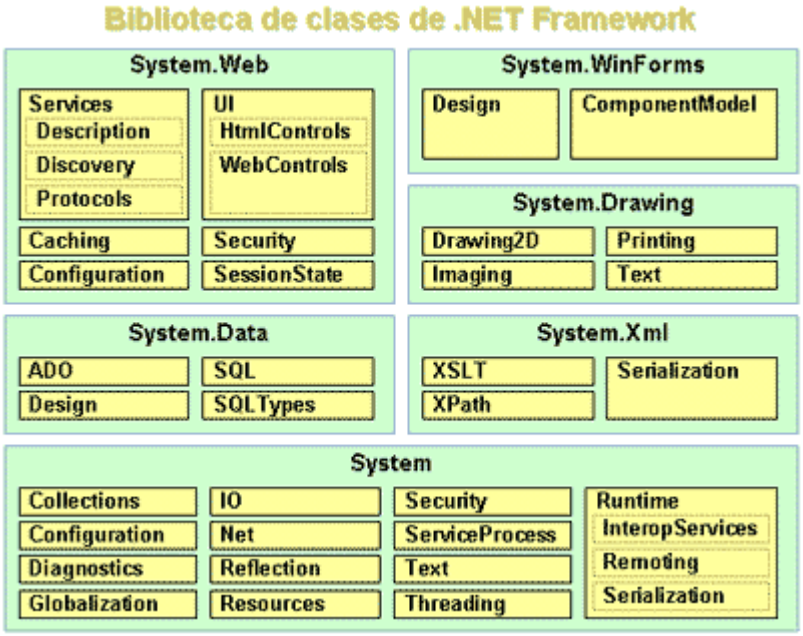

Figura 1 Biblioteca de clases de .NET

La biblioteca de clases de .Net Framework incluye, entre otros, tres componentes clave:

- ASP.NET para construir aplicaciones y servicios Web.
- Windows Forms para desarrollar interfaces de usuario.
- ADO.NET para conectar las aplicaciones a bases de datos.

La forma de organizar la biblioteca de clases de .Net dentro del código es a través de los espacios de nombres (namespaces), donde cada clase está organizada en namespace según su funcionalidad. Por ejemplo, para manejar ficheros se utiliza el namespaces System.IO y si lo que se quiere es obtener información de una fuente de datos se utilizará el namespaces System.Data. [23]

1.4.8- Common Language Runtime (CLR)

El CLR es el verdadero núcleo del Framework de .Net, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios que ofrece el sistema operativo estándar Win32.

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un mismo código, denominado código intermedio (MSIL, Microsoft Intermediate Lenguaje). Para generar dicho código el compilador se basa en el Common Language Specification (CLS) que determina las reglas necesarias para crear código MSIL compatible con el CLR.

De esta forma, indistintamente de la herramienta de desarrollo utilizada y del lenguaje elegido, el código generado es siempre el mismo, ya que el MSIL es el único lenguaje que entiende directamente el CLR. Este código es transparente al desarrollo de la aplicación ya que lo genera automáticamente el compilador.

Sin embargo, el código generado en MSIL no es código máquina y por tanto no puede ejecutarse directamente. Se necesita un segundo paso en el que una herramienta denominada compilador JIT (Just-In-Time) genera el código máquina real que se ejecuta en la plataforma que tenga la computadora. De esta forma se consigue con .Net cierta independencia de la plataforma, ya que cada plataforma puede tener su compilador JIT y crear su propio código máquina a partir del código MSIL.

La compilación JIT la realiza el CLR a medida que se invocan los métodos en el programa y el código ejecutable obtenido, se almacena en la memoria caché de la computadora, siendo recompilado sólo cuando se produce algún cambio en el código fuente. [24]

1.4.9-Navegadores

Un navegador Web es una aplicación que le permite a un usuario recuperar y visualizar documentos de hipertexto, que pueden estar escritos en HTML desde servidores Web de todo el mundo a través de Internet. Actualmente los navegadores actuales permiten mostrar o ejecutar gráficos, secuencias de videos, sonido, animaciones así como otros programas además de texto e hipervínculos. La comunicación entre un servidor Web y el navegador se realizan mediante el protocolo HTTP, aunque también soportan FTP y HTTPS (Esta es una versión cifrada del HTTP). [25]

Existen disímiles navegadores entre los que se destacan: **Internet Explorer**, **Mozilla Firefox**, **Opera**, **Koqueror**, **Nestcape Navigator**, entre otros muchos. En Cuba y más específicamente dentro de la universidad los más usados son: Internet Explorer y el Mozilla Firefox, este último poco a poco se va imponiendo sobre el clásico Internet Explorer debido a las grandes ventajas que presenta.

Estudiar los navegadores Web a la hora de crear una aplicación Web es muy necesario porque la aplicación debe ser compatible con la mayor cantidad de navegadores disponibles para pueda ser accesada por cualquier tipo de usuario.

1.5 – Herramientas de desarrollo.

1.5.1 – Visual Studio 2005

Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones Web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C++, Visual C# y Visual J# utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Dichos lenguajes aprovechan las funciones de .NET Framework, que ofrecen acceso a tecnologías claves para simplificar el desarrollo de aplicaciones Web ASP y Servicios Web XML. [26]

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

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

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

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

1.5.2 – SharpDevelop

SharpDevelop es un entorno integrado de desarrollo libre para los lenguajes de programación C# y Visual Basic .NET. Es usado típicamente por aquellos programadores de los citados lenguajes, que no desean o no pueden usar el entorno de desarrollo de Microsoft, el Microsoft Visual Studio.

Hay disponible un port para Mono/Gtk#, llamado MonoDevelop, el cual funciona en otros sistemas operativos. Para el completado automático de código, la aplicación incorpora sus propios parsers. La versión 1.1 de la aplicación puede importar proyectos de Visual Studio .NET. La versión 2.1 ya es capaz de editarlos directamente. [27]

Es un proyecto que comenzó desde los comienzos del lenguaje C# en el año 2000, y cada vez se pone mejor, en su última versión, incorpora soporte C#, Visual Basic .NET, ASP.NET, XML. Es de diseño visual al igual que el Visual Studio, y bastante similar, es compatible con Net Framework 1.1, 2.0, Compact Framework y Mono (el mismo de Linux), viene con debugger, configuración de proyectos, además de ser escrito enteramente en C#.

SharpDevelop incorpora un diseñador de Windows forms, completado de código. Soporta el uso de la combinación de teclas Ctrl + Espacio, depurador incorporado, herramientas para "Ir a Definición", "Encontrar referencias" y "renombrado", escritura de código C#, ASP.NET, ADO.NET, XML y HTML, llaves inteligentes en la escritura de código.

1.6-Framework.

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

Un framework representa una arquitectura de *software* que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Los Frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

Un equipo cuando usa Apache Struts para desarrollar un sitio Web de un banco puede enfocarse en cómo los retiros de ahorros va a funcionar en lugar de preocuparse de cómo se controla la navegación entre las páginas en una forma libre de errores.

Sin embargo, hay quejas comunes acerca de que el uso de frameworks añade código innecesario y que la preponderancia de frameworks competitivos y complementarios significa que el tiempo que se pasaba programando y diseñando ahora se gasta en aprender a usar frameworks.

Fuera de las aplicaciones en la informática, un framework puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo. Es el esqueleto sobre el cual varios objetos son integrados para una solución dada. [28]

1.6.1-.Net

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

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

Esta nueva tecnología de Microsoft ofrece soluciones a los problemas de programación actuales, como son la administración de código o la programación para Internet. Para aprovechar al máximo las características de .Net es necesario entender la arquitectura básica en la que está implementada esta tecnología y así beneficiarse de todas las características que ofrece esta nueva plataforma.

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Los principales componentes de este entorno son:

- Lenguajes de compilación
- Biblioteca de clases de .Net
- CLR (Common Language Runtime) [29]

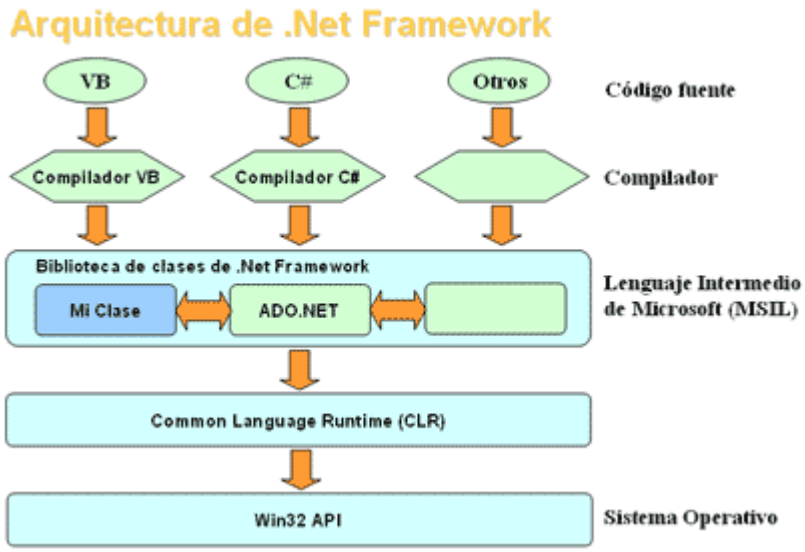

Figura 2 Arquitectura del Framework .NET

1.6.2-Mono

Por definición, Mono es una plataforma de desarrollo de código abierto basada en el .NET Framework. Es una plataforma para ejecutar y desarrollar aplicaciones modernas basadas en los estándares ECMA/ISO. Mono puede ejecutar aplicaciones hechas para los frameworks .NET y Java.

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

Acoge varias licencias libres (X11, LGPL y GPL) y tiene una comunidad de desarrolladores activa, está esponsorizado por Novell y es la base para muchas aplicaciones. Mono incluye compiladores, un intérprete compatible con el CLR de ECMA y un conjunto de librerías. Las librerías cubren la compatibilidad con Microsoft .NET (incluyendo ADO.NET, System.Windows.Forms y ASP.NET).

Mono posee librerías adicionales y otras de terceras partes. Para el desarrollo de aplicaciones gráficas se incluye la librería GTK# que es un "binding" sobre GTK+ y GNOME para permitir el desarrollo de aplicaciones nativas para GNOME utilizando Mono y cualquiera de los lenguajes soportados.

Actualmente Mono se puede ejecutar en plataformas x86, PPC, SPARC y S390 en 32 bits; y x86-64 y SPARC en 64 bits; siendo posible crear y ejecutar aplicaciones en los sistemas operativos: Linux, Windows, OSX, BSD y Solaris. [30]

1.7-Tecnología y lenguajes a usar en la solución del problema en cuestión.

Para dar solución al problema y después de que se hayan analizado un grupo de tecnologías y lenguajes candidatos, se decidió por arquitectos del sistema, el uso de la plataforma .NET y por ende el lenguaje de desarrollo del lado del servidor ASP.NET, como lenguaje de desarrollo del lado del cliente se usará JavaScript.

Se usará también la tecnología AJAX y para el desarrollo se usará la herramienta de desarrollo Visual Studio 2005, también se usara C# como lenguaje de programación y como gestor de base de datos se usará PostgreSQL.

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

1.8 – conclusiones.

En este capítulo se analizo distintas tecnología y otros sistemas existentes para conocer acerca de las tendencias a nivel mundial y nacional para realizar aplicaciones similares a la que se espera implementar en el transcurso del trabajo, fue vista también la decisión por parte de los arquitectos del sistema de las herramientas y tecnologías a usar para el desarrollo del trabajo.

Capítulo 2

2.1-Introducción

En este capítulo se realizará el análisis propuesto para la solución del problema. Se analizarán posibles implementaciones así como módulos o componentes ya existentes los cuales pueden ser utilizados nuevamente así como de las estrategias de integración del sistema. Se harán descripciones de las clases los tipos de datos a usar en la solución del problema además de dar una descripción de las clases u operaciones que se implementen para dar solución al tema tratado.

2.2-Valoración crítica del diseño propuesto por el analista.

El diseño propuesto por el analista fue valorado como bueno, fácil de entender, las clases del diseño están bien representadas ya que desde él, se pueden obtener las clases fundamentales que deben ser definidas para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, dando una idea clara de lo que se debe implementar.

Unido a esto se encuentran los diagramas de interacción, que explican la colaboración que existe entre las clases y cómo son llamados los métodos y sentencias dentro de cada una, de los cuales se obtiene la información necesaria para conocer el orden de las acciones a implementar. Mientras más legible sea el diseño propuesto más fácil es la adaptación de los programadores al mismo y por ende se agiliza el proceso de traducción de clases del diseño a clases de implementación.

Una característica del diseño que evidencia la claridad del trabajo es el uso de grupos de patrones de diseño como es el caso de los “GRASP” y los “GOF”. Los patrones “GRASP” se utilizan con el objetivo de asignar responsabilidades a las diferentes clases que se definen en el diseño. Dentro de este grupo se identifican cinco patrones muy utilizados: experto, creador, alta cohesión, bajo acoplamiento y el controlador.

Estos patrones se les aplican a las clases definidas en el diseño, distribuyendo responsabilidades entre las mismas de forma tal que no existan muchas relaciones, que no se sobrecargue de métodos a una clase en específico pudiendo acomodarlos en otras, entre otras mejoras que brinda el uso de este grupo de patrones.

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

Del gran número de patrones propuestos por GOF, se propone el uso de los patrones “Abstract factory” y “Factory method”, estos patrones tienen como principal función crear objetos sin importar en que momento y a que clase pertenecen.

El uso de este último permitirá crear entidades del negocio de una forma mas rápida, fácil e independiente debido a las funciones básicas para insertar, seleccionar, actualizar y eliminar se agruparían en un subsistema llamado “Abstrac Factory”, con este subsistema se comunicaría una nueva clase llamada “Repositorio”, la cual se encargaría de gestionar todos estos procesos en relación con las entidades del negocio y mediante ella las clases del negocio accederían a estos procesos para dar respuesta a la interfaz de usuario.

2.3-Análisis de posibles implementaciones, componentes o módulos ya existentes que puedan ser reusados.

Para dar solución al problema propuesto en el trabajo es necesario hacer uso de otros módulos que se encuentran ya implementados como es el caso del módulo “Configuración” el cual es fundamental a la hora de hacer búsquedas, ya que en este módulo es donde se trabaja con informaciones básicas referente a los productos como es el caso de: unidad de medida, vía de administración, datos estos necesarios a la hora de insertar los datos de un producto por la importancia que tiene el uso correcto de estas propiedades.

Un módulo importante pero que es poco utilizado en la solución, es el de “Inscripción-Admisión”, es poco utilizado porque sólo será necesario en el caso de solicitar una prescripción o dispensar productos de una

prescripción, para los procesos relacionados con la prescripción es necesario buscar el paciente al que se le va a indicar los medicamentos o buscar el paciente al que se le ha asignado dicha prescripción a la hora de hacer la entrega.

Otro módulo que se utilizará es el de NpgSQL, este módulo es primordial a la hora de establecer una comunicación con una base de datos postgresSQL.

NpgSQL: es un .Net Data Provider para el servidor de bases de datos PostgreSQL. Este permite a una aplicación .Net (Console, WinForms, ASP.NET, Web Services...) usar un servidor PostgreSQL para mandar y recibir datos. Ha sido desarrollado usando las especificaciones de la documentación de .Net. El uso de este módulo es muy importante ya que hace más fácil y rápido el proceso de conexión a la base de datos PostgresSQL.

Un módulo importante para la implementación es **Health Level 7 (HL7)**, esta es una organización internacional, iniciada en los Estados Unidos en 1987, que pretende promover el desarrollo y evolución del estándar HL7 para el formato de datos e intercambio de información entre diferentes sistemas de información de salud. [31]

Se trata de una iniciativa que comenzó en 1987, en base a la necesidad de normalizar las interfaces entre los múltiples sistemas heterogéneos de información, y rápidamente se convirtió en el estándar de facto para el intercambio electrónico de datos clínicos y administrativos en los servicios de salud de los Estados Unidos.

El propósito de HL7 es permitir el intercambio y la integración de los datos que provienen del proceso de la atención médica, a través del desarrollo de guías, metodologías y servicios en general, ofreciendo interoperabilidad entre sistemas de información en salud, de manera flexible y eficiente en cuanto a costos.

Esto lo logra debido a que:

- Permite el intercambio de información entre aplicaciones desarrolladas por diferentes proveedores de *software*.
- Reduce el trabajo en papel, mejorando el soporte a las decisiones y permitiendo la integración de la información de salud, entre diferentes servicios.
- Permite la conectividad entre sistemas heterogéneos a costos competitivos.

- Ofrece flexibilidad, porque puede implementarse usando diversas tecnologías de *software*.
- Reduce los recursos invertidos en la integración entre aplicaciones.
- Reduce los recursos invertidos en programación y mantenimiento de interfaces propietarias.

Figura 3 Descripción de HL7

El Hermes es una implementación confiable del estándar HL7 en su versión 2.3.1, desarrollada por un proyecto del área temática de la Facultad 7 perteneciente a la UCI. Dicho framework en la próxima versión contendrá un conjunto de herramientas que permitirán la administración, integridad y flujo de la información médica entre los distintos Sistemas de Información para la Salud (SIS).

Este garantiza un fuerte tipado, con notaciones más familiares al usuario y equivalentes a los tipos encapsulados en su núcleo, el usuario no necesita conocer las especificaciones en detalle de HL7 para utilizar el mismo en sus aplicaciones. Las herramientas que brinda este módulo posibilitan el envío y recepción de datos clínicos, información de pacientes y algunos reportes de laboratorio, entre los distintos SIS que lo utilicen, empleando para ello un sólo formato, el especificado por HL7.

Reduce los recursos invertidos en la negociación de las interfaces entre aplicaciones para la salud. Con su utilización se lograría incrementar el prestigio del sistema de salud cubano, así como la confianza internacional en el mismo. Permite el ahorro de una suma importante de dinero al país, al evitar la compra de implementaciones del estándar a empresas extranjeras.

2.4-Estrategias de integración.

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

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

2.5-Descripción de los algoritmos no triviales a implementar.

Uno de los procesos fundamentales que ocurren en la farmacia hospitalaria es la “Dispensación de medicamentos”, este proceso está siempre visible en el momento en que llega un pedido de medicamentos, ya sea desde una sala o de otro cualquier servicio del hospital así como de una prescripción de medicamentos (receta médica).

Cuando el farmacéutico recibe un pedido, debe seleccionar cuidadosamente los productos que va a suministrar, este proceso es un poco complicado debido a que para cada producto es posible que existan varias tarjetas de control ya que estos productos se organizan por lotes y fechas de vencimientos, es decir que cada lote de productos tiene una fecha de vencimiento, por lo que el farmacéutico debe seleccionar cuidadosamente el lote con el que va a despachar el medicamento guiándose por la fecha de vencimiento.

Para resolver este problema fue necesario implementar un algoritmo en el cual el farmacéutico no tenga que preocuparse para escoger el lote, ya que la aplicación se encargaría de hacer la distribución necesaria y el farmacéutico sólo debe preocuparse por ver si existe la cantidad de productos necesaria

para hacer la distribución. Esta cantidad (existencia total) esta dada por la suma de la existencia en tarjeta de cada lote.

El algoritmo se encargaría de buscar entre todas las tarjetas de control de cada producto del pedido la que esté más próxima a vencerse y de ahí escoger la cantidad necesaria para entregar, en el caso de existir dos o más tarjetas con la misma fecha de vencimiento, estas serían organizadas por la existencia en tarjeta, ya que mientras menor sea la existencia de productos de esa tarjeta, es necesario agotar primero esa cantidad para que no existan tarjetas con poca existencia innecesariamente.

Si la cantidad del pedido es mayor que la existencia en tarjeta el algoritmo despacha esa cantidad con la que tiene en esa tarjeta y buscaría la próxima en la lista previamente organizada, en donde quedan descartadas las tarjetas en la cual la existencia sea cero para así evitar información innecesaria.

El algoritmo se encargaría también de actualizar la existencia total del producto, así como de insertar en la tabla de movimiento la salida del producto, ya que es necesario tener controlado cada movimiento, para saber de dónde se han extraído los productos para servir el pedido.

2.5.1- Análisis de complejidad del algoritmo.

Para conocer la complejidad del algoritmo es necesario calcular la complejidad ciclomática del mismo, para hacer dicho cálculo es necesario primero tener el código o el diseño del algoritmo, luego enmarcar cada instrucción del código con un número, que representa cada lugar del camino que puede seguir la secuencia del algoritmo, a continuación se representa el código con sus instrucciones enmarcadas.

```
public bool AsignarIdControl(Int32 idDocumento,String codigoCompleto,Int32 cantidad)
       { 1
          AsignacionProductoViewRepositorio asignacionProducto = new AsignacionProductoViewRepositorio(); 1
          AsignacionProductoView asignar = new AsignacionProductoView(); 1
          SolicitudProducto solicit = new SolicitudProducto(); 1
          asignar.CodigoCompleto = codigoCompleto; 1
          int cant = asignacionProducto.ObtenerTodos(asignar).Count; 1
          int cont = 0; 1
          List<AsignacionProductoView> lista = asignacionProducto.ObtenerTodos(asignar); 1
          SolicitudMovimiento relacion = new SolicitudMovimiento(); 1
          ControlProducto control = new ControlProducto(); 1
          SolicitudMovimientoRepositorio Solicitudes = new SolicitudMovimientoRepositorio(); 1
          ParametroProducto tempParametro = new ParametroProducto(); 1
          GestionarTargetaControlNegocio targeta = new GestionarTargetaControlNegocio(); 1
          MovimientoProductoRepositorio movimientos = new MovimientoProductoRepositorio();  1
```

```
 MovimientoProducto movimiento = new MovimientoProducto(); 1
 UsuarioSistemaRepositorio usuarios = new UsuarioSistemaRepositorio();  1
 UsuarioSistema usuario = new UsuarioSistema();  1
 Int32 cantServida = cantidad; 1
 usuario.CodigoUsuario = 1;  1
 targeta.actualizarExist(codigoCompleto);  1
 tempParametro.CodigoCompleto = codigoCompleto;  1
 solicit.CodigoCompleto = codigoCompleto;  1
 solicit.IdDocSalida = idDocumento;  1
if (lista.Count > 0) 2
  {

      while (cantidad > 0) 3
      {
          movimiento.CodigoCompleto = codigoCompleto; 4
          movimiento.FechaMovimiento = DateTime.Now;  4
          movimiento.IdMovimiento = movimientos.ObtenerTodos().Count + 1;  4
          movimiento.IdTipoMovimiento = 1;  4
          movimiento.Observaciones = " ";  4
          movimiento.UsuarioDistribuye = usuarios.ObtenerUno(usuario).NombreUsuario; 4
          movimiento.UsuarioRecibe = " "; 4
          relacion.CodigoCompleto = codigoCompleto; 4
          relacion.IdDocSalida = idDocumento; 4
          relacion.IdMovimiento = movimiento.IdMovimiento; 4
          control.IdControlMedicamento = lista[cont].IdControlProducto; 4
          control.CodigoCompleto = codigoCompleto; 4
          control = targeta.ObtenerTargeta(control); 4
          if (lista[cont].ExistenciaTarjeta > cantidad) 5
          {
              movimiento.Cantidad = cantidad; 6
              movimiento.IdControlMedicamento = lista[cont].IdControlProducto;  6
              control.ExistenciaTarjeta = control.ExistenciaTarjeta - cantidad;  6
              cantidad = 0;  6


          }
          else
              if (lista[cont].ExistenciaTarjeta == cantidad) 7
              {
                  movimiento.Cantidad = cantidad; 8
                  movimiento.IdControlMedicamento = lista[cont].IdControlProducto;  8
                  control.ExistenciaTarjeta = 0;  8
                  cantidad = 0;  8
              }
              else
              {
                  cantidad = cantidad - lista[cont].ExistenciaTarjeta.Value; 9
                  movimiento.Cantidad = lista[cont].ExistenciaTarjeta;  9
                  control.ExistenciaTarjeta = 0;  9

              }

          cont++; 10
```

```
                    cantServida = cantServida - cantidad; 10

movimiento.Saldo = Int32.Parse((parametroProducto.ObtenerUno(tempParametro).ExistTotal).ToString()) -
cantServida; 10
                    targeta.ModificarTargeta(control); 10
                    targeta.actualizarExist(codigoCompleto); 10
                    movimientos.Adicionar(ref movimiento); 10
                    Solicitudes.Adicionar(ref relacion); 10
                } 11


                }

                solicit = productoSalida.ObtenerUno(solicit); 12
                solicit.CantidadServida = cantServida; 12
                productoSalida.Actualizar(solicit); 12

            return true;  12
        } 13
```

Después de este paso, es necesario representar el grafo de flujo asociado, en el cual se representan distintos componentes como es el caso de:

Nodo: son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.

Regiones: son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

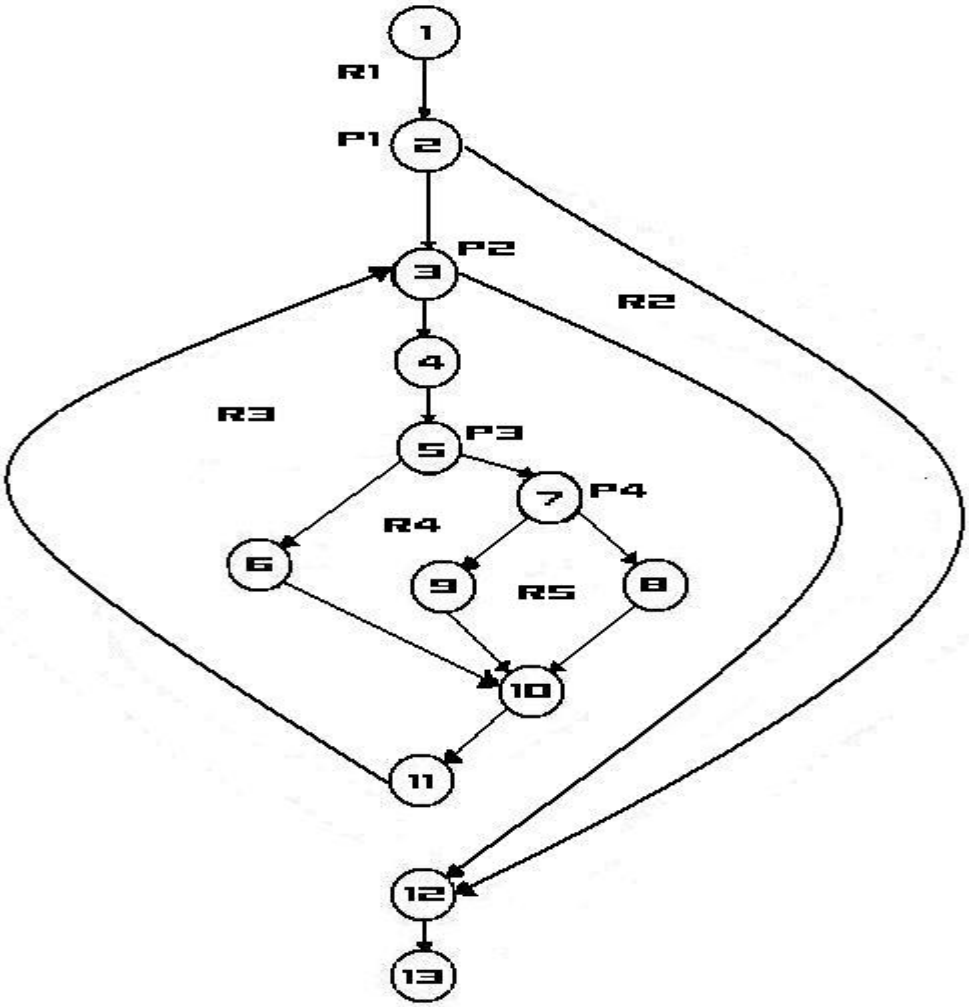

Figura 4 Grafo de flujo del algoritmo.

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

V (G) = (A – N) + 2

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

V (G) = (16 – 13) + 2
V (G) = 5

V (G) = P + 1
Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).
V (G) = 4 +1
V (G) = 5

V (G) = R
Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del calculo.
V (G) = 5

Realizado el cálculo por las 3 vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática de 5 que da la visión de que existen a lo sumo cinco caminos lógicos por donde recorrer el algoritmo.

2.6-Estructura de datos a usar para implementar el algoritmo.

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

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

La lista permite dado el valor de una posición obtener el elemento que se encuentra en la misma sin tener que recorrer innecesariamente toda la lista para obtener el valor necesario, posibilita insertar un objeto en la posición deseada desplazando los demás objetos a la posición inmediata inferior, también permite

eliminar un objeto, adicionar un objeto, además que muestra mediante la propiedad “Count” la cantidad de objetos que contiene así como se puede saber si tiene o no elementos.

2.7-Estándares de código.

Los estándares de codificación son reglas específicas a una lengua que reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores. Los estándares de codificación no destapan problemas existentes, evitan más bien que los errores ocurran.

Durante el desarrollo, los estándares de codificación ayudan a los ingenieros a producir un código de alta calidad y a entender y a utilizar el código de sus compañeros de trabajo. Pero también realzan considerablemente la capacidad de mantenimiento y reutilización a largo plazo del producto final.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico.

Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de *software*, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada.

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

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de *software*. La mantenibilidad del código es la facilidad con que el sistema de *software* puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento.

Aunque la legibilidad y la mantenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. El

mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas.

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del *software* y para obtener un buen rendimiento.

Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas y posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de *software* se convierta en un sistema de *software* fácil de comprender y de mantener.

Aunque el propósito principal para llevar a cabo revisiones del código a lo largo de todo el desarrollo es localizar defectos en el mismo, las revisiones también pueden afianzar los estándares de codificación de manera uniforme.

La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final del proyecto de software. No es práctico, ni prudente, imponer un estándar de codificación una vez iniciado el trabajo.

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

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

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

Capítulo 2.

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

2.7.1-Notación Camello.

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

Ejemplo:

public GestionarProducto producto o public GestionarProducto productoNuevo

2.7.2-Notación Pascal

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

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

Espacios de nombres (namespaces).

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

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

Clases.

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

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

Public partial class ProductoFarmacia
{
     //….
}
```

Métodos

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

```
public void RegistrarTargeta (ControlMedicamentos entidad)
{
     //…
}
```

Propiedades (Properties)

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

```
public String NombreProducto
{
   //…
```

```
}
```

Componentes.

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

- TextBoxNombre
- ButtonAceptar
- DropDownListFechaVencimiento
- GridViewListadoProducto

Estructura del código.

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

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

Ejemplo:

```
public Int32 IdProducto
{
    get
    {
        return this.idProducto;
    }
    set
    {
```

```
                    this.idProducto = value;
                }
            }
```

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

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

Ejemplo:

```
for (int i = 0; i < GridViewListadoProducto.Rows.Count; i++)
{
    if (GridViewListadoProducto.Rows[ i ].Cells[ 3 ].Text  ==  “Activo”)
    {
        //….
    }
}
```

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

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

2.8-Descripción de las nuevas clases u operaciones implementadas

2.8.1-Entidades del negocio

Nombre: RelProdPedido	
Tipo de clase: entidad	
Atributo	**Tipo**
idPedido	Int32
cantidadPedida	Int32
cantidadServida	Int32
codigoCompleto	String
Para cada responsabilidad:	
Nombre:	**Descripción:**
IdPedido	Propiedad para mostrar y modificar el ID del pedido
CantidadPedida	Propiedad para mostrar y modificar la cantidad pedida
CantidadServida	Propiedad para mostrar y modificar la cantidad servida
CodigoCompleto	Propiedad para mostrar y modificar código del producto

Tabla 1 Clase entidad "RelProdPedido".

Nombre: Proveedor	
Tipo de clase: entidad	
Atributo	**Tipo**
idProveedor	Int32
descProveedor	String
direccion	String
telefono	String
Para cada responsabilidad:	

Nombre:	**Descripción:**
Proveedor	Es el constructor de la clase.
CodigoProveedor	Propiedad para mostrar y modificar el código del proveedor.
DescProveedor	Propiedad para mostrar y modificar la descripción del proveedor.
Direccion	Propiedad para mostrar y modificar la dirección del proveedor.
Telefono	Propiedad para mostrar y modificar el teléfono del proveedor.

Tabla 2 Clase entidad "Proveedor".

Nombre: LibroFechaVencimiento	
Tipo de clase: entidad	
Atributo	**Tipo**
fechaVencimiento	DateTime
alertaVencimiento	Int32
codigoCompleto	String
idControlMedicamento	Int32
Para cada responsabilidad:	
Nombre:	**Descripción:**
LibroFechaVencimiento	Constructor de la clase
FechaVencimiento	Propiedad para mostrar y modificar la fecha de vencimiento.
CodigoCompleto	Propiedad para mostrar y modificar el código del producto.
AlertaVencimiento	Propiedad para mostrar y modificar la alerta de vencimiento.
IdControlMedicamento	Propiedad para mostrar y modificar el ID del control de medicamentos.

Tabla 3 Clase entidad "LibroFechaVencimiento".

Nombre: DocumentoEntrada	
Tipo de clase: entidad	
Atributo	**Tipo**
idDocEntrada	Int32
codigoUsuario	Int32
idProveedor	Int32
idTipoMovimiento	Int32
descDocEntrada	String
fechaEntrada	DateTime
observaciones	String
Para cada responsabilidad:	
Nombre:	**Descripción:**
IdDocEntrada	Propiedad para mostrar y modificar el ID del documento de entrada.
CodigoUsuario	Propiedad para mostrar y modificar el código del usuario que envía el documento
IdProveedor	Propiedad para mostrar y modificar el ID del proveedor.
IdTipoMovimiento	Propiedad para mostrar y modificar el ID del movimiento que representa el documento
DescDocEntrada	Propiedad para mostrar y modificar la descripción del documento
FechaEntrada	Propiedad para mostrar y modificar la fecha del documento
Observaciones	Propiedad para mostrar y modificar las observaciones sobre el documento

Tabla 4 Clase Entidad "DocumentoEntrada".

Nombre: DocumentoSolicitud	
Tipo de clase: entidad	
Atributo	**Tipo**
idDocSalida	Int32
codigoUsuario	Int32
idServicio	Int32
descDocSalida	String
fechaSalida	DateTime
usuarioSolicita	String
usuarioRecibe	String
observaciones	String
idEstadoSolicitud	Int32
idTipoMovimiento	Int32
Para cada responsabilidad:	
Nombre:	**Descripción:**
IdDocSalida	Propiedad para mostrar y modificar el ID del documento de salida.
CodigoUsuario	Propiedad para mostrar y modificar el código del usuario.
IdServicio	Propiedad para mostrar y modificar el ID del servicio que solicita.
DescDocSalida	Propiedad para mostrar y modificar la descripción del documento.
FechaSalida	Propiedad para mostrar y modificar la fecha de salida del documento.
UsuarioSolicita	Propiedad para mostrar y modificar el nombre del usuario que solicita el producto.
UsuarioRecibe	Propiedad para mostrar y modificar el usuario que recibe el producto.
Observaciones	Propiedad para mostrar y modificar las observaciones del documento.

IdEstadoSolicitud	Propiedad para mostrar y modificar el ID del estado del documento.
IdTipoMovimiento	Propiedad para mostrar y modificar el ID del tipo del movimiento.

Tabla 5 Clase entidad "DocumentoSolicitud".

Nombre: ControlProductos	
Tipo de clase: entidad	
Atributo	**Tipo**
idControlMedicamento	Int32
codigoSector	String
idFuenteRecurso	Int32
numeroEstiba	Int32
seccion	String
estante	String
fechaRecibo	DateTime
existenciaTerjeta	Int32
observaciones	String
fechaVencimiento	DateTime
loteEntrada	String
codigoCompleto	String
Para cada responsabilidad:	
Nombre:	**Descripción:**
ControlMedicamentos	Constructor de la clase.
IdControlMedicamento	Propiedad para mostrar y modificar el ID del control de medicamentos.
CodigoSector	Propiedad para mostrar y modificar el código del sector de donde viene el medicamento.
IdFuenteRecurso	Propiedad para mostrar y modificar el ID de la fuente de

	recurso.
NumeroEstiba	Propiedad para mostrar y modificar el número de estiba.
Seccion	Propiedad para mostrar y modificar la sección de almacenamiento.
Estante	Propiedad para mostrar y modificar el estante donde se almacena.
FechaRecibo	Propiedad para mostrar y modificar la fecha que se recibió el producto.
Existencia	Propiedad para mostrar y modificar la existencia del producto.
Observaciones	Propiedad para mostrar y modificar las observaciones.
FechaVencimiento	Propiedad para mostrar y modificar la fecha de vencimiento.
LoteEntrada	Propiedad para mostrar y modificar el lote de entrada.
CodigoCompleto	Propiedad para mostrar y modificar el código del producto.

Tabla 6 Clase entidad " ControlMedicamentos".

Nombre: RelControlProdInventario	
Tipo de clase: entidad	
Atributo	**Tipo**
idInventario	Int32
idControlMedicamento	Int32
cantidadRegistrada	Int32
cantidadFisReal	Int32
auditoria	Boolean
codigoCompleto	Int32
Para cada responsabilidad:	
Nombre:	**Descripción**
ControlProdInventario	Constructor de la clase.
IdInventario	Propiedad para mostrar y modificar el ID del inventario.
IdControlMedicamento	Propiedad para mostrar y modificar el ID del control del

	producto.
CantidadRegistrada	Propiedad para mostrar y modificar la cantidad registrada.
CantidadFisReal	Propiedad para mostrar y modificar la cantidad física del producto.
Auditoria	Propiedad para mostrar y modificar el dato de auditoria.
CodigoCompleto	Propiedad para mostrar y modificar el código del producto.

Tabla 7 Clase entidad " RelControlProdInventario".

Nombre: DevolucionProducto	
Tipo de clase: entidad	
Atributo	**Tipo**
idDevolucion	Int32
idServicio	Int32
codigoUsuario	Int32
fechaEntrada	DateTime
fechaSalida	DateTime
Para cada responsabilidad:	
Nombre:	**Descripción:**
DevolucionProducto	Constructor de la clase.
IdDevolucion	Propiedad para mostrar y modificar el ID de la devolución.
IdOrigen	Propiedad para mostrar y modificar el ID del origen de la devolución.
CodigoUsuario	Propiedad para mostrar y modificar el código del usuario.
FechaEntrada	Propiedad para mostrar y modificar la fecha de entrada.
FechaSalida	Propiedad para mostrar y modificar la fecha de salida.

Tabla 8 Clase entidad " DevolucionProducto".

Nombre: Inventario	
Tipo de clase: entidad	
Atributo	**Tipo**
idInventario	Int32
nombreInventario	String
fechaInventario	DateTime
observaciones	String
codigoUsuario	Int32
Para cada responsabilidad:	
Nombre:	**Descripción:**
Inventario	Constructor de la clase.
IdInventario	Propiedad para mostrar y modificar el ID del inventario.
NombreInventario	Propiedad para mostrar y modificar el nombre del inventario.
FechaInventario	Propiedad para mostrar y modificar la fecha del inventario.
Observaciones	Propiedad para mostrar y modificar las observaciones.
CodigoUsuario	Propiedad para mostrar y modificar el código del usuario.

Tabla 9 Clase entidad " Inventario".

Nombre: MovimientoProducto	
Tipo de clase: entidad	
Atributo	**Tipo**
idMovimiento	Int32
codigoCompleto	Int32
fechaMovimiento	DateTime
usuarioDistribuye	String
usuarioRecibe	String
cantidad	Int32
observaciones	String
saldo	Int32

idControlMedicamento	Int32
idTipoMovimiento	Int32
Para cada responsabilidad:	
Nombre:	**Descripción:**
IdMovimiento	Propiedad para mostrar y modificar el ID del movimiento
CodigoCompleto	Propiedad para mostrar y modificar el código del producto
FechaMovimiento	Propiedad para mostrar y modificar la fecha del movimiento
UsuarioDistribuye	Propiedad para mostrar y modificar el usuario que distribuye
UsuarioRecibe	Propiedad para mostrar y modificar el usuario que recibe
Cantidad	Propiedad para mostrar y modificar la cantidad que se mueve
Observaciones	Propiedad para mostrar y modificar las observaciones
Saldo	Propiedad para mostrar y modificar la existencia del producto
IdControlMedicamento	Propiedad para mostrar y modificar el ID del control del medicamento
IdTipoMovimiento:	Propiedad para mostrar y modificar el ID del tipo del movimiento

Tabla 10 Clase entidad "MovimientoProducto".

Nombre: ParametroProducto	
Tipo de clase: entidad	
Atributo	**Tipo**
stockSeguridad	Double
stockMaximo	Double
stockMinimo	Double
loteReposicion	String
diasStock	Int32
existTotal	Double
reserva	Double
dreserva	Double

mediaMensual	Double
ultimaSalida	Double
stockMaxTentativo	Double
consumoFarmacia	Double
mayorConsumo	Double
menorConsumo	Double
mesMayorConsumo	DateTime
mesMenorConsumo	DateTime
codigoCompleto	String
Para cada responsabilidad:	
Nombre:	**Descripción:**
ParametroProducto	Constructor de la clase.
StockSeguridad	Propiedad para mostrar y modificar el stock de seguridad.
StockMaximo	Propiedad para mostrar y modificar el stock máximo.
StockMinimo	Propiedad para mostrar y modificar el stock mínimo.
LoteReposicion	Propiedad para mostrar y modificar el lote de reposición.
DiasStock	Propiedad para mostrar y modificar los días comprendidos del stock.
ExistTotal	Propiedad para mostrar y modificar la existencia total del producto.
Reserva	Propiedad para mostrar y modificar la reservar.
Dreserva	Propiedad para mostrar y modificar la doble reserva.
MediaMensual	Propiedad para mostrar y modificar la media mensual.
UltimaSalida	Propiedad para mostrar y modificar la cantidad de la última salida.
StockMaxTentativo	Propiedad para mostrar y modificar el stock máximo tentativo.
ConsumoFarmacia	Propiedad para mostrar y modificar el consumo de la farmacia.
MayorConsumo	Propiedad para mostrar y modificar el mayor consumo.
MenorConsumo	Propiedad para mostrar y modificar el menor consumo.
MesMayorConsumo	Propiedad para mostrar y modificar el mes de mayor consumo

MesMenorConsumo	Propiedad para mostrar y modificar el mes de menor consumo
IdProducto	Propiedad para mostrar y modificar el ID del producto.

Tabla 11 Clase entidad " ParametroProducto".

Nombre: Pedidos	
Tipo de clase: entidad	
Atributo:	**Tipo:**
idPedido	Int32
codigoUsuario	Int32
fechaPedido	DateTime
idProveedor	String
idEstadoPedido	Int32
Para cada responsabilidad:	
Nombre:	**Descripción:**
Pedidos	Constructor de la clase
IdPedido	Propiedad para mostrar y modificar el ID del pedido.
CodigoUsuario	Propiedad para mostrar y modificar el código del usuario.
FechaPedido	Propiedad para mostrar y modificar la fecha del pedido.
IdProveedor	Propiedad para mostrar y modificar el ID del proveedor.
IdEstado	Propiedad para mostrar y modificar el ID del estado.

Tabla 12 Clase entidad "Pedidos".

Nombre: PrescripcionPaciente	
Tipo de clase: entidad	
Atributo	**Tipo**
numPrescripcion	Int32
fechaPrescripcion	DateTime
fechaEntregaMedicamento	DateTime

usernameMedico	String
idPersona	Int32
codigoCompleto	String
cantidadSolicitada	Int32
dosis	String
cantidadServida	Int32
idServicio	Int32
Para cada responsabilidad: que va a hacer la clase	
Nombre:	**Descripción:**
PrescripcionPaciente	Constructor de la clase
NumPrescripcion	Propiedad para mostrar y modificar el número de la prescripción.
FechaPrescripcion	Propiedad para mostrar y modificar la fecha de la prescripción.
FechaEntregaMedicamento	Propiedad para mostrar y modificar la fecha de entrega de los medicamentos.
UsernameMedico	Propiedad para mostrar y modificar el nombre del medico.
IdPersona	Propiedad para mostrar y modificar el ID de la persona.
CodigoCompleto	Propiedad para mostrar y modificar el código del producto.
Dosis	Propiedad para mostrar y modificar la dosis prescripta.
CantidadServida	Propiedad para mostrar y modificar la cantidad servida.
CantidadSolicitada	Propiedad para mostrar y modificar la cantidad solicitada.

Tabla 13 Clase entidad "PrescripcionPaciente".

Nombre: RelProductoDocEntrada	
Tipo de clase	
Atributo	**Tipo**
idDocEntrada	Int32
cantidad	Int32
fechaVencimiento	DateTime

codigoCompleto	String
Para cada responsabilidad: que va a hacer la clase	
Nombre:	**Descripción:**
ProductoDocEntrada	Constructor de la clase.
IdDocEntrada	Propiedad para mostrar y modificar el ID del documento de entrada.
Cantidad	Propiedad para mostrar y modificar la cantidad del producto.
FechaVencimiento	Propiedad para mostrar y modificar la fecha de vencimiento del producto.
CodigoCompleto	Propiedad para mostrar y modificar el código del producto.

Tabla 14 Clase entidad "RelProductoDocEntrada".

Nombre: ProductoFarmacia	
Tipo de clase	
Atributo	**Tipo**
idEstado	Int32
codigoProd	String
idConcentracion	Int32
idViaAdmin	Int32
observaciones	String
codigoGrupoProducto	String
idFormaFarmaceutica	Int32
idUnidadMedida	Int32
idFormaAlmacenamiento	Int32
idPresentacionProducto	Int32
codigoCompleto	String
idSubgrupoProducto	Int32
Para cada responsabilidad: que va a hacer la clase	

Nombre:	Descripción:
ProductoFarmacia	Constructor de la clase.
IdEstado	Propiedad para mostrar y modificar el ID.
CodigoProd	Propiedad para mostrar y modificar el código del producto.
IdConcentracion	Propiedad para mostrar y modificar el ID de la concentración.
IdViaAdmin	Propiedad para mostrar y modificar el ID de la vía de administración.
Observaciones	Propiedad para mostrar y modificar las observaciones.
CodigoGrupoProducto	Propiedad para mostrar y modificar el código del grupo del producto.
IdFormaFarmaceutica	Propiedad para mostrar y modificar el ID de la forma farmacéutica.
IdUnidadMedida	Propiedad para mostrar y modificar el ID de la unidad de medida.
IdFormaAlmacenamiento	Propiedad para mostrar y modificar el ID de la forma de almacenamiento.
IdPresentacionProducto	Propiedad para mostrar y modificar el ID de la presentación del producto.
CodigoCompleto	Propiedad para mostrar y modificar el ID del producto
IdSubgrupoProducto	Propiedad para mostrar y modificar el ID del sub grupo de productos.

Tabla 15 Clase entidad "ProductoFarmacia".

Nombre: RelSolicitudProducto	
Tipo de clase entidad	
Atributo	**Tipo**
idControlMedicamento	Int32
idDocSalida	Int32
cantidadServida	Int32

cantidadSolicitada	Int32
codigoCompleto	Int32
Para cada responsabilidad: que va a hacer la clase	
Nombre:	**Descripción:**
RelSolicitudProducto	Constructor de la clase.
IdDocSalida	Propiedad para mostrar y modificar el ID del documento de salida.
CantidadServida	Propiedad para mostrar y modificar la cantidad servida.
CantidadSolicitada	Propiedad para mostrar y modificar la cantidad solicitada.
CodigoCompleto	Propiedad para mostrar y modificar el ID del producto.

Tabla 16 Clase entidad " RelSolicitudProducto".

Nombre: SectorFarmacia	
Tipo de clase entidad	
Atributo	**Tipo**
codigoSector	String
idTipoSector	Int32
descSector	String
Para cada responsabilidad: que va a hacer la clase	
Nombre:	**Descripción:**
SectorFarmacia	Constructor de la clase.
CodigoSector	Propiedad para mostrar y modificar el código del sector.
IdTipoSector	Propiedad para mostrar y modificar el ID del tipo de sector.
DescSector	Propiedad para mostrar y modificar la descripción del sector.

Tabla 17 Clase entidad "SectorFarmacia"

2.8.2-Clases Controladoras

Nombre: GestionarPedido	
Tipo de clase: controladora	
Atributo	**Tipo**
pedidos	PedidosRepositorio
stock	ParametroProductoRepositorio
productoPedido	ProdPedidoRepositorio
Para cada responsabilidad:	
Nombre:	**Descripción:**
RegistrarPedidos	Método que sirve para insertar pedidos.
ModificarPedidos	Método para modificar los pedidos.
EliminarPedidos	Método para eliminar pedidos.
ObtenerPedidos	Método para obtener un pedido.
ObtenerPedidosTodos	Método para obtener todos los pedidos de la base de datos.

Tabla 18 Clase controladora "GestionarPedido".

Nombre: GestionarDevolucion	
Tipo de clase: Controladora	
Atributo	**Tipo**
repositorio	DevolucionProductoRepositorio
Para cada responsabilidad:	
Nombre:	**Descripción:**
ObtenerUnaDevolucion	Método para obtener una devolución de un producto determinado.
ObtenerTodasDevoluciones	Método para obtener un listado de devoluciones.
AdicionarDevolucion	Método para adicionar devoluciones.
ActualizarDevolucion	Método para actualizar devoluciones.
EliminarDevolucion	Método para eliminar devoluciones.

Tabla 19 Clase controlador "GestionarDevolucion".

Nombre: GestionarEntrada	
Tipo de clase controladora	
Atributo	**Tipo**
entrada	DocumentoEntradaRepositorio
tarjeta	ControlMedicamentosRepositorio
movimiento	MovimientoProductoRepositorio
pedidos	PedidosRepositorio
docEntrada	ProductoDocEntradaRepositorio
Para cada responsabilidad:	
Nombre:	**Descripción:**
RegistrarEntrada	Método para insertar una entrada de productos.
ModificarEntrada	Método para actualizar o modificar una entrada de productos.
EliminarEntrada	Método para eliminar una entrada de productos.
ObtenerEntrada	Método para obtener una entrada de productos determinados.
ObtenerEntradasTodas	Método para obtener un listado de entradas de productos.
RegistrarTarjeta	Método para insertar una tarjeta de control de medicamentos.
InsertarMovimiento	Método para insertar un movimiento de productos.
ObtenerPedido	Método para obtener un pedido de productos determinado
ModificarPedidos	Método para modificar un pedido de productos.
RegistrarProdEntrada	Método para insertar una relación producto – entrada.
ActualizarProdEntrada	Método para actualizar una relación producto – entrada.
ElminarProdEntrada	Método para eliminar una relación producto – entrada.
ObtenerUnDocEntrada	Método para obtener los datos de un documento de entrada.
ObtenerTodosDocEntrada	Método para obtener los datos de un listado de documentos de entrada.

Tabla 20 Clase controladora "GestionarEntrada"

Nombre: GestionarEntrega	
Tipo de clase: controladora	
Atributo	**Tipo**
solicitud	DocumentoSolicitudRepositorio
tarjeta	ControlMedicamentosRepositorio
movimiento	MovimientoProductoRepositorio
Para cada responsabilidad:	
Nombre:	**Descripción:**
ObtenerSolicitud	Método para obtener una solicitud determinada de productos.
ModificarSolicitud	Método para modificar una solicitud de productos.
InsertarMovimiento	Método para insertar un movimiento de productos.

Tabla 21 Clase controladora "GestionarEntrega"

Nombre: GestionarPrescripcionPaciente	
Tipo de clase: controladora	
Atributo	**Tipo**
repositorio	PrescripcionPacienteRepositorio
Para cada responsabilidad:	
Nombre:	**Descripción:**
ObtenerUnoPrescripcion	Método para obtener una prescripción determinada de productos.
ObtenerTodasPrescripciones	Método para obtener un listado de prescripciones.
Adicionar	Método para adicionar una prescripción de productos.
Actualizar	Método para actualizar una prescripción de productos.
Eliminar	Método para eliminar una prescripción de productos.

Tabla 22 Clase controladora "GestionarPrescripcion"

Nombre: GestionarProducto	
Tipo de clase: controladora	
Atributo	**Tipo**
productoFarmacia	ProductoFarmaciaRepositorio
Para cada responsabilidad:	
Nombre:	**Descripción:**
InsertarProducto	Método para insertar un producto.
ObtenerTodos	Método para obtener un listado de productos.
ActualizarProducto	Método para actualizar los datos de un producto.
EliminarProducto	Método para eliminar un producto.
ObtenerUnProducto	Método para obtener un producto determinado.

Tabla 23 Clase controladora "GestionarProducto"

Nombre: GestionarSolicitud	
Tipo de clase: controladora	
Atributo	**Tipo**
solicitud	DocumentoSolicitudRepositorio
prescripcion	PrescripcionPacienteRepositorio
productoSalida	ProductoSalidaRepositorio
Para cada responsabilidad:	
Nombre:	**Descripción:**
RegistrarSolicitud	Método para registrar una solicitud de productos.
ModificarSolicitud	Método para modificar una solicitud de productos.
EliminarSolicitud	Método para eliminar una solicitud de productos.
ObtenerSolicitud	Método para obtener una solicitud determinada de productos.
ObtenerSolicitudTodas	Método para obtener un listado de solicitudes de productos.
ObtenerPrescripcion	Método para obtener una prescripción de productos.

RegistrarProdSolicitud	Método para registrar una entrada de la relación producto - solicitud.
EliminarProdSolicitud	Método para eliminar una relación solicitud - producto.
ActualizarProdSolicitud	Método para actualizar una relación producto – solicitud.
ObtenerProSolicitud	Método para obtener una relación determinadas de producto – solicitud.
ObtenerTodosProdSolicitud	Método para obtener un listado de relaciones producto - solicitud.
AsignarIdControl	Método que ejecuta el algoritmo para distribuir los productos según el pedido.
DispensarProductos	Método para hacer la distribución de productos para las prescripciones.

Tabla 24 Clase controladora "GestionarSolicitud"

Nombre: GestionarTarjetaControl	
Tipo de clase: controladora	
Atributo	**Tipo**
tarjeta	ControlMedicamentosRepositorio
movimiento	MovimientoProductoRepositorio
Para cada responsabilidad:	
Nombre:	**Descripción:**
RegistrarTarjeta	Método para registrar los datos de una tarjeta de control de un producto.
ModificarTarjeta	Método para modificar los datos de una tarjeta de control de un producto.
EliminarTarjeta	Método para eliminar una tarjeta de control de un producto.
ObtenerTarjeta	Método para obtener los datos de una tarjeta de control determinada

ObtenerTarjetasTodas	Método para obtener un listado de tarjetas de control.
RegistrarMovimiento	Método para registrar un movimiento de productos.
ActualizarExist	Método para actualizar la existencia de un producto determinado.
ExistenciaTotal	Método para actualizar la existencia de un producto determinado después de efectuarse un movimiento.
VerVencimientos	Método para buscar los productos que estén próximos a vencerse.
ProductosVencidos	Método para buscar los productos que se hayan vencido.

Tabla 25 Clase controladora "GestionarTarjetaControl"

Nombre: GestionarStock	
Tipo de clase: controladora	
Atributo	**Tipo**
repositorio	ParametroProductoRepositorio
Para cada responsabilidad:	
Nombre:	**Descripción:**
ObtenerUno	Método para obtener los datos de stock de un producto.
ObtenerTodos	Método sobrecargado para obtener un listado de datos de varios productos.
Adicionar	Método para adicionar los datos de stock de un producto.
Actualizar	Método para actualizar los datos de stock de un producto.
Eliminar	Método para eliminar los datos de stock de un producto.
CalcularConsumoAnual	Método para calcular el consumo anual de un producto.
MayorConsumo	Método para calcular el mayor consumo de un producto.
MenorConsumo	Método para calcular el menor consumo de un producto.
MesMayorConsumo	Método para calcular el mes de mayor consumo.
MesMenorConsumo	Método para calcular el mes de menor consumo.
Stock	Método para hacer el cálculo básico de stock.

StockMaximo	Método para calcular el stock máximo de un producto.
StockMinimo	Método para calcular el stock mínimo de un producto.

Tabla 26 Clase controladora "GestionarStock".

Nombre: GestionarMovimiento	
Tipo de clase: controladora	
Atributo	**Tipo**
repositorio	MovimientoProductoRepositorio
Para cada responsabilidad:	
Nombre:	**Descripción:**
ObtenerUno	Método para devolver los movimientos del lote de un producto determinado.
ObtenerTodos	Método sobrecargado para devolver un listado de movimientos para uno o varios lotes de un producto determinado.
Adicionar	Método para adicionar movimientos de un lote de producto.
Actualizar	Método para actualizar los movimientos de un lote de productos.
Eliminar	Método para eliminar los movimientos de un lote de producto.
BuscarMovimientrosRangos	Método para obtener una lista de movimientos para un lote de productos en un rango de fechas determinado.

Tabla 27 Clase controladora "GestionarMovimiento"

2.8.3- Factory

Las factory son clases que se implementan para acceder a la base de datos de una forma más fácil, estas clases pertenecen al patrón de diseño de fabricación pura, las mismas heredan de interfaces (son clases implementadas en C# para sustituir la herencia múltiple) por lo cual para implantarlas es necesario redefinir los métodos que traen definidos dichas interfaces.

Estas ejecutan llamadas a los procedimientos de almacenado de la base de datos para hacer esta comunicación invisible al programador que sólo debe preocuparse por implementar los métodos necesarios para acceder a ellas.

Son un paquete de cinco clases por cada tabla de la base de datos, cada una de estas clases tiene la función de ejecutar un procedimiento de almacenado distinto. Estas clases son accesadas a través de una clase repositorio, que es como tal la clase que ejecuta los métodos necesarios para llamar a las factory, es necesario implementar un repositorio para cada una de las clases.

La implementación de esta clase representa una pequeña capa entre las capas de negocio y acceso a datos en la arquitectura en tres capas para realizar la unión entre estas capas. Debido al carácter repetitivo de estas clases, sólo se mostraran las descripciones de las clases implementadas para una entidad de la base de datos.

Nombre: PedidosFactory	
Tipo de clase: Factory	
Atributo	**Tipo**
entity	Pedidos
Para cada responsabilidad:	
Nombre:	**Descripción:**
Construct	Método para construir el objeto de la entidad pedido

Tabla 28 PedidosFactory

Nombre: PedidosInsertFactory	
Tipo de clase: InsertFactory	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Execute	Método sobrecargado que ejecuta la llamada al procedimiento de almacenado “Insert” de la base de datos.

Tabla 29 PedidosInsertFactory

Nombre: PedidosUpdateFactory	
Tipo de clase: UpdateFactory	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Execute	Método sobrecargado que ejecuta la llamada al procedimiento de almacenado “Update” de la base de datos.

Tabla 30 PedidosUpdateFactory

Nombre: PedidosDeleteFactory	
Tipo de clase: DeleteFactory	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Execute	Método sobrecargado que ejecuta la llamada al procedimiento de almacenado “Delete” de la base de datos.

Tabla 31 PedidosDeleteFactory

Nombre: PedidosSelectionFactory	
Tipo de clase: SelectionFactory	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Execute	Método sobrecargado que ejecuta la llamada al procedimiento de almacenado “Select” de la base de datos.

Tabla 32 PedidosSelectionFactory

Nombre: PedidosRepositorio	
Tipo de clase: Repositorio	
Atributo	**Tipo**
factory	IDomainObjectFactory<Pedidos>
Para cada responsabilidad:	
Nombre:	**Descripción:**
ObtenerUno	Método que haciendo una llamada a una SelectionFactory devuelve una entidad del objeto pedido.
ObtenerTodos	Método que haciendo una llamada a una SelectionFactory devuelve una lista de entidades del objeto pedido.
Adicionar	Método que haciendo una llamada a un Insertfactory inserta los datos de un pedido en la base de datos.
Actualizar	Método que haciendo una llamada a un Updatefactory actualiza los datos de un pedido en la base de datos.
Eliminar	Método que haciendo una llamada a un Deletefactory elimina los datos de un pedido en la base de datos.

Tabla 33 PedidosRepositorio

2.8.4- Clases Interfaces

Nombre: InsertarProductos	
Tipo de clase: interface	
Atributo	**Tipo**
Para cada responsabilidad:	

Nombre:	Descripción:
LimpiarPagina	Ejecuta instrucciones para limpiar la página.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
DropDownListGrupo_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
DropDownListSubGrupo_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
TextBoxFormaFarmA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxFormaFarmB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxFormaAlmacA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxFormaAlmacB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxUnidadMedA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxUnidadMedB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxConcA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxConcB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxViaAdminA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxViaAdminB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxPrecA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.

TextBoxPrecB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
ButtonGuardar_Click	Ejecuta las instrucciones para insertar el producto.
ButtonLimpiar_Click	Realiza un llamado a la función “limpiar pagina”.

Tabla 34 CI "InsertarProductos”.

Ver anexos figura 6.

Nombre: ActualizarProductos	
Tipo de clase: interface	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
LimpiarPagina	Ejecuta instrucciones para limpiar la página.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
DropDownListGrupo_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
DropDownListSubGrupo_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
DropDownListNombre_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
TextBoxFormaFarmA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxFormaFarmB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxFormaAlmacA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.

TextBoxFormaAlmacB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxUnidadMedA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxUnidadMedB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxConcA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxConcB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxViaAdminA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxViaAdminB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxPrecA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxPrecB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
ButtonGuardar_Click	Ejecuta las instrucciones para insertar el producto.
ButtonCambiarNombre_Click	Muestra los componentes necesarios para cambiar el nombre del producto
ButtonLimpiar_Click	Realiza un llamado a la función “limpiar pagina”.

Tabla 35 CI "ActualizarProductos”.

Ver anexos figura 7.

Nombre: EliminarProductos	
Tipo de clase: interface	
Atributo	**Tipo**

Para cada responsabilidad:	
Nombre:	**Descripción:**
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
DropDownListGrupo_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
DropDownListSubGrupo_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
DropDownListNombre_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
ButtonEliminar_Click	Ejecuta las instrucciones para eliminar un producto.

Tabla 36 CI "EliminarProductos".

Ver anexos figura 8.

Nombre: EntrarProductos	
Tipo de clase: interface	
Atributo	**Tipo**
ProductoLista	List<ProductoFarmaciaView>
ProductoListaBus	List<ProductoFarmaciaView>
Para cada responsabilidad:	
Nombre:	**Descripción:**
LimpiarPagina	Ejecuta instrucciones para limpiar la página.
LimpiarBusqueda	Ejecuta instrucciones para limpiar la búsqueda.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
ButtonIniciarBus_Click	Ejecuta las instrucciones para realizar la

	búsqueda.
DropDownListGrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
DropDownListSubgrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del subgrupo de productos.
DropDownListNombreBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del nombre del producto.
ButtonAdicionarBus_Click	Ejecuta las instrucciones para adicionar los productos al listado de productos.
ButtonSalvar_Click	Ejecuta las instrucciones para guardar y enviar los datos de entrada de un producto.
GridViewListado_PageIndexChanging	Ejecuta instrucciones para el paginado del gridview del listado de productos.
GridViewResultBus_PageIndexChanging	Ejecuta instrucciones para el paginado del gridview para los resultados de la búsqueda.
GridViewListado_SelectedIndexChanged	Ejecuta instrucciones cuando cambia las selección de listado de productos
TextBoxPrecentProdA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxPrecentProdB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.

Tabla 37 Cl "EntrarProductos".

Ver anexos figura 9.

Nombre: SolicitarProductos	
Tipo de clase: interface	
Atributo	**Tipo**
ProductoLista	List<ProductoFarmaciaView>
ProductoListaBus	List<ProductoFarmaciaView>
Para cada responsabilidad:	
Nombre:	**Descripción:**
LimpiarPagina	Ejecuta instrucciones para limpiar la página.
LimpiarBusqueda	Ejecuta instrucciones para limpiar la búsqueda.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
ButtonIniciarBus_Click	Ejecuta las instrucciones para realizar la búsqueda.
DropDownListGrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
DropDownListSubgrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del subgrupo de productos.
DropDownListNombreBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del nombre del producto.
ButtonAdicionarBus_Click	Ejecuta las instrucciones para adicionar los productos al listado de productos.
ButtonGuardar_Click	Ejecuta las instrucciones para guardar y enviar los datos de entrada de un producto.
GridViewListado_PageIndexChanging	Ejecuta instrucciones para el paginado del gridview del listado de productos.
GridViewResultBus_PageIndexChanging	Ejecuta instrucciones para el paginado del

	gridview para los resultados de la búsqueda.
GridViewListado_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección del listado de productos.
TextBoxPrecentProdA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxPrecentProdB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.

Tabla 38 CI "SolicitarProductos".

Ver anexos figura 10.

Nombre: BandejaSolicitudes	
Tipo de clase: interface	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
GridViewListadoSolicitudes_SelectedIndexChanged	Ejecuta instrucciones cuando cambia las selección de listado de productos
ButtonEntrega_Click	Ejecuta las instrucciones para realizar la entrega de los productos del listado.

Tabla 39 CI "BandejaSolicitudes".

Ver anexos figura 11.

Nombre: Pedidos	
Tipo de clase: interface	
Atributo	**Tipo**
ProductoLista	List<ProductoFarmaciaView>
ProductoListaBus	List<ProductoFarmaciaView>
Para cada responsabilidad:	
Nombre:	**Descripción:**
LimpiarPagina	Ejecuta instrucciones para limpiar la página.
LimpiarBusqueda	Ejecuta instrucciones para limpiar la búsqueda.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
ButtonIniciarBus_Click	Ejecuta las instrucciones para realizar la búsqueda.
DropDownListGrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
DropDownListSubgrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del subgrupo de productos.
DropDownListNombreBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del nombre del producto.
ButtonAdicionarBus_Click	Ejecuta las instrucciones para adicionar los productos al listado de productos.
ButtonSolicitar	Ejecuta las instrucciones para realizar la solicitud de productos.
DropDownListProveedor_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del proveedor.
GridViewListado_PageIndexChanging	Ejecuta instrucciones para el paginado del

GridViewResultBus_PageIndexChanging	gridview del listado de productos. Ejecuta instrucciones para el paginado del gridview para los resultados de la búsqueda.

Tabla 40 CI "Pedidos".

Ver anexos figura 12.

Nombre: EntradasPedido	
Tipo de clase: interface	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
GridViewPedidos_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección del gridview del listado de pedidos.
ButtonGuardar_Click	Ejecuta las intrusiones para realizar la entrada de los productos del pedido.

Tabla 41 CI "EntradasPedidos".

Ver anexos figura 13.

Nombre: VerDisponibilidad	
Tipo de clase: interface	
Atributo	**Tipo**
Para cada responsabilidad:	

Nombre:	**Descripción:**
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
ButtonIniciarBus_Click	Ejecuta las instrucciones para realizar la búsqueda.
DropDownListGrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
DropDownListSubgrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del subgrupo de productos.
DropDownListNombreBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del nombre del producto.

Tabla 42 CI "VerDisponibilidad".

Ver anexos figura 14

Nombre: Inventario	
Tipo de clase: interface	
Atributo	**Tipo**
ProductoLista	List<ProductoFarmaciaView>
ProductoListaBus	List<ProductoFarmaciaView>
Para cada responsabilidad:	
Nombre:	**Descripción:**
LimpiarPagina	Ejecuta instrucciones para limpiar la página.
LimpiarBusqueda	Ejecuta instrucciones para limpiar la búsqueda.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.

ButtonIniciarBus_Click	Ejecuta las instrucciones para realizar la búsqueda.
DropDownListGrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del grupo de productos.
DropDownListSubgrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del subgrupo de productos.
DropDownListNombreBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownList del nombre del producto.
ButtonAdicionarBus_Click	Ejecuta las instrucciones para adicionar los productos al listado de productos.
ButtonGuardar_Click	Ejecuta las instrucciones para guardar los datos necesarios de inventario.
GridViewListado_PageIndexChanging	Ejecuta instrucciones para el paginado del gridview del listado de productos.
GridViewResultBus_PageIndexChanging	Ejecuta instrucciones para el paginado del gridview para los resultados de la búsqueda.
TextBoxPrecentProdA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxPrecentProdB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.

Tabla 43 CI "Inventario".

Ver anexos figura15.

Nombre: DatosConsumo	
Tipo de clase: interface	
Atributo	**Tipo**
Producto	ParametroProducto
ProductoListaBus	List<ProductoFarmaciaView>
Para cada responsabilidad:	
Nombre:	**Descripción:**
LimpiarPagina	Ejecuta instrucciones para limpiar la página.
LimpiarBusqueda	Ejecuta instrucciones para limpiar la búsqueda.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
ButtonIniciarBus_Click	Ejecuta las instrucciones para realizar la búsqueda.
DropDownListGrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del grupo de productos.
DropDownListSubgrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del subgrupo de productos.
DropDownListNombreBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del nombre del producto.
ButtonAdicionarBus_Click	Ejecuta las instrucciones para adicionar los productos al listado de productos.
ButtonGuardar_Click	Ejecuta las instrucciones para guardar los datos de consumo
ButtonCalcularFechas_Click	Ejecuta instrucciones para realiza los cálculos de consumos correspondiente para el producto seleccionado

GridViewResultBus_PageIndexChanging	Ejecuta instrucciones para el paginado del gridview para los resultados de la búsqueda.
TextBoxPrecentProdA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxPrecentProdB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.

Tabla 44 CI "DatosConsumo".

Ver anexos figura 16.

Nombre: ControlStock	
Tipo de clase: interface	
Atributo	**Tipo**
Producto	ParametroProducto
ProductoListaBus	List<ProductoFarmaciaView>
ProductoLista	List<StockProducto>
Para cada responsabilidad:	
Nombre:	**Descripción:**
LimpiarPagina	Ejecuta instrucciones para limpiar la página.
LimpiarBusqueda	Ejecuta instrucciones para limpiar la búsqueda.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
ButtonIniciarBus_Click	Ejecuta las instrucciones para realizar la búsqueda.
DropDownListGrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del grupo de productos.
DropDownListSubgrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la

	selección de dropdownlist del subgrupo de productos.
DropDownListNombreBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del nombre del producto.
ButtonAdicionarBus_Click	Ejecuta las instrucciones para adicionar los productos al listado de productos.
ButtonCalcularDias_Click	Ejecuta instrucciones para realiza calcular la cantidad de días que existen entre dos fechas.
ButtonCalcular_Click	Ejecuta las instrucciones para realizar los cálculos de stock.
GridViewListado_PageIndexChanging	Ejecuta instrucciones para el paginado del gridview del listado de productos.
GridViewResultBus_PageIndexChanging	Ejecuta instrucciones para el paginado del gridview para los resultados de la búsqueda.
TextBoxPrecentProdA_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.
TextBoxPrecentProdB_TextChanged	Ejecuta instrucciones cuando cambia el texto del textbox.

Tabla 45 CI "ControlStock".

Ver anexos figura17.

Nombre: TarjetaEstiba	
Tipo de clase: interface	
Atributo	**Tipo**
ListadoTarjetas ProductoListaBus	List<RelacionProductoControlView> List<ProductoFarmaciaView>
Para cada responsabilidad:	
Nombre:	**Descripción:**

LimpiarPagina	Ejecuta instrucciones para limpiar la página.
LimpiarBusqueda	Ejecuta instrucciones para limpiar la búsqueda.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
ButtonIniciarBus_Click	Ejecuta las instrucciones para realizar la búsqueda.
DropDownListGrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del grupo de productos.
DropDownListSubgrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del subgrupo de productos.
DropDownListNombreBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del nombre del producto.
ButtonAdicionarBus_Click	Ejecuta las instrucciones para adicionar los productos al listado de productos.
ButtonSalvar_Click	Ejecuta las instrucciones para guardar los datos de control del producto.
ButtonEliminar_Click	Ejecuta las instrucciones para eliminar una tarjeta de control.
GridViewListadoTarjeta_SelectedIndexChanged	Ejecuta las instrucciones para cuando cambia la tarjeta seleccionada.

Tabla 46 Cl "TarjetaEstiba".

Ver anexos figura 18.

Nombre: SolicitarPrescripcion	
Tipo de clase: interface	
Atributo	**Tipo**
Persona	DatosPersonas
ProductoLista	List<ProductoFarmaciaView>
ListaPersonaBus	List<DatosPersonas>
ProductoListaBus	List<ProductoFarmaciaView>
Para cada responsabilidad:	
Nombre:	**Descripción:**
LimpiarPagina	Ejecuta instrucciones para limpiar la página.
LimpiarBusqueda	Ejecuta instrucciones para limpiar la búsqueda.
LimpiarBusPersona	Limpia la búsqueda de personas.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
ButtonIniciarBus_Click	Ejecuta las instrucciones para realizar la búsqueda.
DropDownListGrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del grupo de productos.
DropDownListSubgrupoBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del subgrupo de productos.
DropDownListNombreBus_SelectedIndexChanged	Ejecuta instrucciones cuando cambia la selección de dropdownlist del nombre del producto.
ButtonAdicionarBus_Click	Ejecuta las instrucciones para adicionar los productos al listado de productos.
ButtonBuscarBus_Click	Realiza la búsqueda de personas.
ButtonSeleccionarBus_Click	Para seleccionar un paciente

ButtonAceptar_Click	Guarda los datos de la prescripción de productos.

Tabla 47 CI "SolicitarPrescripcion".

Ver anexos figura19.

Nombre: Dispensario	
Tipo de clase: interface	
Atributo	**Tipo**
ProductoLista	List<ProductoFarmaciaView>
Persona	DatosPersonas
Para cada responsabilidad:	
Nombre:	**Descripción:**
LimpiarPagina	Ejecuta instrucciones para limpiar la página.
LimpiarBusPersona	Limpia la búsqueda de personas.
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
ButtonBuscarBus_Click	Realiza la búsqueda de personas.
ButtonSeleccionarBus_Click	Para seleccionar un paciente
ButtonAceptar_Click	Realiza el despacho de productos.

Tabla 48 CI "Dispensario".

Ver anexos figura 20.

Nombre: Vencimientos	
Tipo de clase: interface	
Atributo	**Tipo**
ProductosVencidos	List<RelacionProductoControlView>
ProductosPorVencerse	List<RelacionProductoControlView>
Para cada responsabilidad:	

Nombre:	Descripción:
Page_Load	Ejecuta instrucciones cuando se esta cargando la página.
ComponerFechas	Construye una fecha valida.
RestarFechas	Resta dos fechas.

Tabla 49 Cl "Vencimientos".

Ver anexos figura 21.

2.9-Conclusiones.

El trabajo en este capítulo, tiene gran importancia debido a que se hace una descripción de las clases fundamentales en el desarrollo, lo que posibilitará futuros mantenimientos de la aplicación. Fue muy importante dar a conocer los otros módulos, con los cuales se vinculaba la solución propuesta. Así como el estándar de código usado, por su importancia en la organización y entendimiento del trabajo realizado.

La descripción del algoritmo no trivial fue de gran ayuda para dar a conocer como fue resuelto el problema para la distribución de los productos para cada solicitud y no menos importante fue la explicación del uso de la estructura de datos usada dado por los beneficios que brinda la lista genérica de C#.

Capítulo 3

3.1 – Introducción.

Desde que el implementador empieza a desarrollar una aplicación, se va encontrando a su paso con posibles errores que de ocurrir pudieran afectar el rendimiento de la aplicación, una forma de ver si la aplicación esta libre de estos errores es testeando el código o las interfaces.

Este capítulo, tratará fundamentalmente sobre los tipos de pruebas que es necesario realizar para que una aplicación sea probada ya sea en tiempo de compilación (cuando se esta compilando el programa) o en tiempo de ejecución (cuando se está ejecutando la aplicación). Además se muestran ejemplos de estos test.

3.2 – Descripción de las pruebas.

Los “test de unidades” son pruebas llevadas a cabo por los implementadores sobre las unidades mínimas desarrolladas por ellos, estas unidades pueden ser clases, métodos, propiedades, componentes, etc. Estas unidades se prueban separadas unas de otras y básicamente se hacen durante la implementación del *software*.

Para realizar estas pruebas es necesario establecer una serie de reglas que sirven como objetivos de estas pruebas:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir errores.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

El objetivo de diseñar pruebas se basa en que sistemáticamente le muestren al implementador diferentes tipos de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

Capítulo 3.

Los “test de unidades” son orientados casi siempre a las pruebas de “caja blanca” aunque para realizar uno de estos test es necesario probar el flujo de datos desde la interfaz del componente. Si los datos no entran correctamente, todas las demás pruebas no tienen sentido. Estas pruebas son aplicadas a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que estos funcionen como se espera.

Las “pruebas de caja blanca” es prácticamente el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del *software* proponiendo casos de pruebas que examinan que estén correctos todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Estos casos de pruebas generan los caminos lógicos o posibles que debe recorrer el flujo de la ejecución, para afirmar que se esta ejecutando correctamente el procedimiento.

Una de las técnicas para ejecutar las pruebas de caja blanca es la del “Camino básico”, el resultado de esta técnica es una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática, define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Esta técnica fue usada anteriormente en este trabajo en el capítulo 2, epígrafe 2.5.1 “Análisis de complejidad del algoritmo”.

Otra forma de probar el código es mediante las pruebas de “caja negra”. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Para fijar estas pautas de diseño de pruebas, se apoya en las siguientes dos definiciones de un caso de prueba bien elegido:

- El que reduce el número de otros casos necesarios para que la prueba sea razonable. Esto implica que el caso ejecute el máximo número de posibilidades de entradas diferentes para así reducir el total de casos.
- Cubre un conjunto extenso de otros casos posibles, es decir, indica algo acerca de la ausencia o la presencia de defectos en el conjunto específico de entradas que prueba, así como de otros conjuntos similares.

Capítulo 3.

Una de las técnicas mas usadas dentro de las pruebas de caja negra es la “Partición de Equivalencia” esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del *software*. Es también una de las más efectivas, pues permite examinar los valores válidos y no válidos de las entradas existentes en el *software*.

Mediante esta técnica se descubren de forma inmediata una serie de errores que de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico, además se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de prueba que hay que desarrollar.

Para definir las clases de equivalencia es necesario tener en cuenta un conjunto de reglas:

- Si una condición de entrada especifica un rango, entonces se confeccionan una clase de equivalencia válida y 2 no válidas.
- Si una condición de entrada especifica la cantidad de valores, identificar una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada especifica un conjunto de valores de entrada y existen razones para creer que el programa trata en forma diferente a cada uno de ellos, identificar una clase válida para cada uno de ellos y una clase no válida.
- Si una condición de entrada especifica una situación de tipo “debe ser”, identificar una clase válida y una no válida.
- Si existe una razón para creer que el programa no trata de forma idéntica ciertos elementos pertenecientes a una clase, dividirla en clases de equivalencia menores.

Luego de tener las clases válidas y no válidas definidas, se procede a definir los casos de pruebas, pero para ello antes se debe haber asignado un identificador único a cada clase de equivalencia. Luego entonces se pueden definir los casos teniendo en cuenta lo siguiente:

- Escribir un nuevo caso que cubra tantas clases de equivalencias válidas no cubiertas como sea posible hasta que todas las clases de equivalencias hayan sido cubiertas por casos de prueba.
- Escribir un nuevo caso de prueba que cubra una y sólo una clase de equivalencia no válida hasta que todas las clases de equivalencias no válidas hayan sido cubiertas por casos de pruebas.

Con la aplicación de esa técnica se obtiene un conjunto de pruebas que reduce el número de casos de pruebas y dice algo sobre la presencia o ausencia de errores. A menudo se plantea que las pruebas a los *software* nunca terminan, simplemente se transfiere del desarrollador al cliente.

Cada vez que el cliente usa el programa está llevando a cabo una prueba. Aplicando el diseño de casos de pruebas al *software* en cuestión se puede conseguir una prueba más completa y descubrir y corregir el mayor número de errores antes de que comiencen las “pruebas del cliente”.

3.3 – Aplicación de pruebas de caja blanca.

Para realizar el test es necesario realizar primeramente los procesos descritos en el capítulo 2, epígrafe 2.5.1 “Análisis de complejidad del algoritmo” para calcular los valores de la complejidad ciclomática del procedimiento al cual se le va a aplicar la prueba.

A continuación se enumera las sentencias de código del procedimiento.

```
public Double ExistenciaTotal(String codigoCompleto,Int32 cantidad,int idTipoMovimiento)

       { 1
        ParametroProductoRepositorio parametro = new ParametroProductoRepositorio();1
        ParametroProducto producto = new ParametroProducto();1
        ParametroProducto temp = new ParametroProducto(); 1
        temp.CodigoCompleto = codigoCompleto; 1
        producto = parametro.ObtenerUno(temp); 1
         if (idTipoMovimiento == 4) 2
          {
              producto.ExistTotal = producto.ExistTotal + cantidad;  3
          }
          else
              if (idTipoMovimiento == 1) 4
              {
                  if ((parametro.ObtenerUno(temp).ExistTotal) >= 0) 5
                      {
                        producto.ExistTotal = producto.ExistTotal - cantidad;  6
                      }
                      else
                      {
```

```
                    producto.ExistTotal = 0; 7
                }
            }
        parametro.Actualizar(producto); 8
        return producto.ExistTotal.Value; 8
    }
```

Seguidamente se construye el grafo de flujo asociado al código anterior.

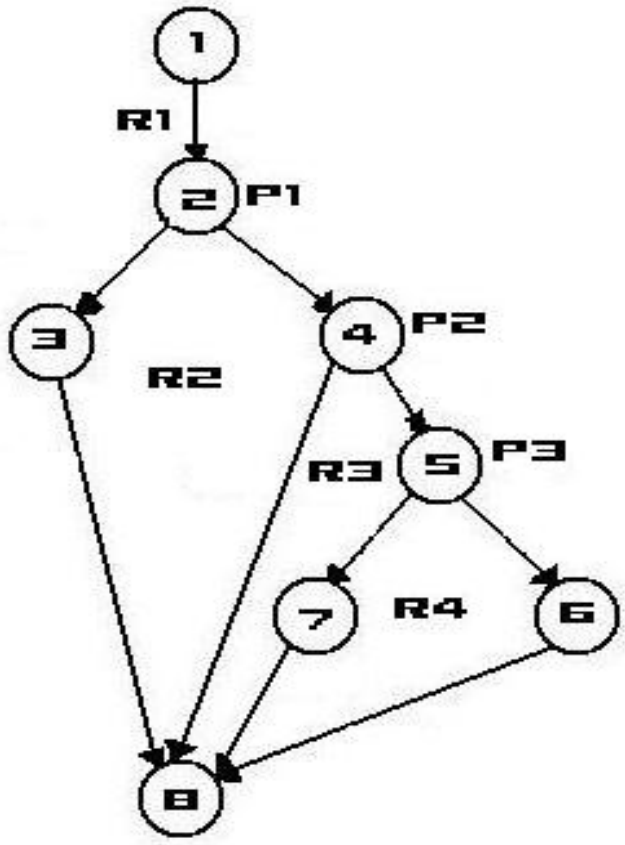

Figura 5 Grafo de flujo para el código anterior.

Luego de haber construido el grafo, se realiza el cálculo de la complejidad ciclomática, mediante las tres fórmulas descritas en el capítulo 2, epígrafe 2.5.1 “Análisis de complejidad del algoritmo”, las cuales tienen que arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

Capítulo 3.

Fórmulas para calcular complejidad ciclomática.

1 – V (G) = (A – N) + 2.

2 – V (G) = P + 1.

3 – V (G) = R.

Aplicando estas fórmulas al grafo de flujo de la figura 5 se obtienen los siguientes resultados:

Calculando mediante la fórmula 1:

V (G) = (10 - 8) + 2

V (G) = 4.

Calculando mediante la fórmula 2:

V (G) = 3 + 1

V (G) = 4.

Calculando mediante la fórmula 3:

V (G) = 4.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 4, lo que significa que existen cuatro posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo. En estas representaciones se subrayan los elementos de cada camino que los hacen independientes a los demás.

Camino básico #1:

1 – 2 – 3 – 8.

Camino básico #2:

1 – 2 – 4 – 8.

Camino básico #3:

1 – 2 – 4 – 5 – 7 – 8.

Camino básico #4:

1 – 2 – 4 – 5 – 6 – 8.

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico, es necesario aclarar que las entradas de los datos al procedimiento tratado puede provenir de varias clases interfaz, en las cuales se define el código, la cantidad y el ID del tipo de movimiento, ya que este procedimiento se usa para calcular el saldo (cantidad resultante después de una entrada o salida) después de hacer cada movimiento.

Por esto fue necesario implementar una pequeña aplicación auxiliar, con una interfaz sencilla que permita hacer la entrada de datos necesarios para poder probar el procedimiento sin tener que acceder a la aplicación original.

Para realizar los casos de pruebas es necesario cumplir con las siguientes exigencias:
Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo,
Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
Entrada: Se muestran los parámetros que entran al procedimiento
Resultados Esperados: Se expone resultado que se espera que devuelva el procedimiento.

Caso de prueba para el camino básico # 1.

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: el código de producto será un string no nulo, la cantidad de entrada será un numero entero entre 0 y 1000, el valor de ID del tipo de movimiento será un numero entero mayor que 0.

Condición de ejecución: el ID del tipo de movimiento será igual a “4” y la cantidad se igual a 200.

Entrada: Código = A0234521, cantidad = 200, ID = 4.

Resultados esperados: Se espera un valor mayor o igual que 200.

Las dos primera ejecuciones del caso de prueba no dieron el resultado que se esperaba debido a que existían errores de codificación, ya que se intercambiaban los datos internamente y no se condicionaba correctamente, el procedimiento, luego de las correcciones pertinentes se obtuvo un resultado de 285 y como se esperaba un resultado mayor o igual que 200 se puede concluir como satisfactorio el caso de prueba.

Caso de prueba para el camino básico # 2:

Descripción: La misma descripción que la del caso de prueba del camino #1.

Condición de ejecución: el ID del tipo de movimiento será igual a “3” y la cantidad se igual a 200.

Entrada: Código = A0234521, cantidad = 200, ID = 3.

Resultados esperados: No se espera ningún cambio.

Con la ejecución del caso de prueba se pudo comprobar que por este camino, el recorrido del flujo se convierte en un camino no necesario, debido a que en el caso que el ID no sea “1” o “4” el procedimiento no produce afectaciones.

Caso de prueba para el camino básico # 3:

Descripción: La misma que la del caso de prueba del camino #1.

Condición de ejecución: el ID del tipo de movimiento será igual a “1” y la cantidad se igual a 200.

Entrada: Código = A0234521, cantidad = 200, ID =1.

Resultados esperados: se espera un resultado igual a 0.

Con la ejecución de este caso de prueba, se pudo corroborar que si la cantidad de entrada va ser mayor que la existencia actual del producto, el procedimiento modifica la existencia haciendo esta 0, y así se evita la entrada de un valor menor que 0 a la base de datos de la aplicación.

Caso de prueba para el camino básico # 4:

Descripción: La misma que la del caso de prueba del camino #1.

Condición de ejecución: el ID del tipo de movimiento será igual a “1” y la cantidad se igual a 50.

Entrada: Código = A0234521, cantidad =50, ID =1.

Resultados esperados: se espera un resultado mayor o igual que 0 y menor o igual que 85.

Luego de la ejecución del caso de prueba se puede afirmar que el resultado de 35 es bueno, debido a que se esperaba un valor entre 0 y 85.

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

3.4 – Aplicación de pruebas de caja negra.

Para la aplicación de este tipo de prueba se usará el caso de uso “RegistrarSolicitud”, este caso de uso esta basado en las gestiones necesarias para realizar un pedido de productos por parte de un servicio hospitalario, que puede ser una sala, un laboratorio o un salón de operaciones, así como los procesos que realiza el farmacéutico para despachar estos productos.

Clases Validas	**Clases no validas**	**Resultados de la prueba.**	**Resultados de la prueba.**	**Observaciones**
El usuario da clic en el botón adicionar y se le muestra un popup para buscar el o los productos que va a adicionar a la solicitud.	El usuario al dar clic en el botón adicionar no se le muestre el popup de búsqueda	Luego de dar clic en el botón el usuario ve el popup de búsqueda.	satisfactorio	La operación se realizo correctamente
El usuario procede a realizar la búsqueda del o de los productos que desea adicionar al listado de productos de la solicitud, para esto selecciona los parámetros de búsqueda y da clic en el botón iniciar.	Al dar clic en el botón de búsqueda no se muestren los resultados de la búsqueda.	El usuario observo el resultado de la búsqueda	satisfactorio	La búsqueda se demoro un poco debido a la velocidad de conexión a la base de datos.

El usuario selecciona los productos o el producto que va a adicionar al listado de búsqueda y al dar clic en el botón adicionar del popup se muestran en el listado los productos seleccionados.	Al dar clic en el botón seleccionar del popup los productos seleccionados no se muestren en el listado de productos	Los productos seleccionados fueron pasados al listado.	satisfactorio	Al adicionar los productos al listado se comprobó que era posible adicionar un producto que ya estuviera en el listado, este pequeño error fue corregido posteriormente.
El usuario se dispone a realizar la solicitud dando clic en el botón solicitar para enviar la petición de productos	Los datos no se envían correctamente.	Los datos no se enviaron.	insatisfactorio	Al quedarse campos vacíos del modelo de solicitud el servidor envió un mensaje de error por este motivo
Repetición de la prueba anterior		Los datos se enviaron correctamente	satisfactorio	Se realizaron las validaciones pertinentes para que no se pudiera enviar los datos de la página si quedan parámetros vacíos.

El farmacéutico al cargar la página “Bandeja de Solicitudes” se le muestre las solicitudes hechas hasta ese momento.	En caso de que existan solicitudes, no se observe el listado.	Se muestran las solicitudes correctamente	satisfactorio	La operación se realizo correctamente
Al seleccionar una solicitud se muestra el listado de productos para esa solicitud.	No se muestran los productos de la solicitud seleccionada.	Se muestran correctamente los productos	Satisfactorio.	La operación se realizo correctamente
El usuario procede a hacer el despacho de productos de la solicitud seleccionada dando clic en el botón entregar.	La solicitud no se despacha correctamente.	La solicitud fue despachada correctamente	Satisfactorio.	La operación se realizo correctamente

Tabla 50 "Prueba de caja negra".

3.5 – Conclusiones.

En este capítulo se abordó acerca de los métodos de pruebas más usados en el desarrollo de la aplicación, ya que las pruebas al *software* constituyen una herramienta más para que tenga la calidad requerida antes de ser entregado al cliente, para que el cliente quede satisfecho con el trabajo realizado.

Conclusiones

Como resultado del trabajo se dió cumplimiento al objetivo propuesto: se logró desarrollar una aplicación que resuelve los principales procesos requeridos en la farmacia hospitalaria utilizando correctamente los patrones de diseño establecidos en el análisis.

Además se dio cumplimiento a las principales tares previstas:

- Fueron muy satisfactorios los estudios realizados acerca de otros sistemas informáticos vinculados con la gestión de la información de la farmacia hospitalaria.
- El estudio de los lenguajes y las tecnologías más usadas a nivel mundial para desarrollar aplicaciones de este tipo fue de gran ayuda por la importancia que tiene para un implementador conocer a fondo acerca del lenguaje que va a utilizar para el desarrollo.
- El estudio de los estándares de codificación fue de gran utilidad en la implementación porque sirvió para aprender nuevos métodos y formas de tener organizado el código para futuros mantenimientos o iteraciones para mejorar la aplicación.
- Estudiar el lenguaje de modelado del análisis fue muy satisfactorio porque mientras más visión tenga el implementador del análisis le resultará más fácil la comprensión del trabajo y podrá avanzar más rápido en la implementación.

Recomendaciones

Se recomienda para próximas iteraciones incluir:

- Implementaciones para imprimir reportes y gestionar devoluciones, que no fueron implementados en el reciente trabajo por no constituir un caso de uso crítico para el sistema.
- Mejoras en la interfaz de usuario de la aplicación, buscando formas más amigables para el cliente.
- Sistema de ayuda y manual de usuario para que los usuario puedan contar con una guía para el uso del sistema realizado en este trabajo.

Estudiar la posibilidad de integrar el sistema, con otros ya implementados en la facultad como es el caso de “Sistema para la planificación de materiales gastables de uso medico”.

Referencias Bibliográficas

1 (UAG), U. A. D. G. Coordinación de Informática Médica 2003. Disponible en: http://cim.uag.mx/historia.html

2 Ídem [1].

3 Ídem [1].

4 BIOCOM SISTEMA MODULAR INTEGRADO PARA PRESTADORES DE SERVICIOS DE SALUD 2007.Disponible en: http://www.biocom.com/sistema/index.html

5 Etron Software s.l. Etron Farmacia, 2007. Disponible en: http://www.etronfarmacia.com

6 SOFTEC Soluciones para la Oficina de Farmacia con Gestifarma 2004. Disponible en: http://www.softec.es/Gestion/Gestifarma.htm

7 VISUAL LIMES, S. L. FARHOS, 2005.Disponible en: http://www.visual-limes.com/pdf/farhos_presentacion.pdf

8 AUTORES, C. CARE2X - El Proyecto, 2004.Disponible en: http://www.care2x.org/

9 Colaboradores de Wikipedia. Cliente-servidor, 2007.Disponible en: http://es.wikipedia.org/w/index.php?title=Cliente-servidor&oldid=9702828

10 VEGAS, J. Desarrollo de Aplicaciones Web 2002.Disponible en: http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node17.html

11 Colaboradores de Wikipedia. Aplicación web, 2007. Disponible en: http://es.wikipedia.org/w/index.php?title=Aplicaci%C3%B3n_web&oldid=9719403

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

13 Colaboradores de Wikipedia. PHP, 2007. Disponible en: http://es.wikipedia.org/w/index.php?title=PHP&oldid=9703577

14 Colaboradores de Wikipedia. Perl, 2007.Disponible en: http://es.wikipedia.org/w/index.php?title=Perl&oldid=9652511

15 ULTRASIST, I. E. S. A. ASP, 2004.Disponible en: http://www.ultrasist.com.mx/tecnologias/asp.htm

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

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

18 Colaboradores de Wikipedia. Internet Information Services , 2007.Disponible en: http://es.wikipedia.org/w/index.php?title=Internet_Information_Services&oldid=9082082.

Referencias Bibliográficas.

19 Colaboradores de Wikipedia. Servidor HTTP Apache, 2007.Disponible en: http://es.wikipedia.org/w/index.php?title=Servidor_HTTP_Apache&oldid=9508937.

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

21 Colaboradores de Wikipedia. ASP.NET, 2007. Disponible en: http://es.wikipedia.org/w/index.php?title=ASP.NET&oldid=9713520

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

23 Ídem [20].

24 Ídem [20].

25 NETWORK-PRESS. Funciones del Navegador 2003. Disponible en: http://www.network-press.org/?navegador_concepto

26 CORPORATION, M. Introducción a Visual Studio. 2007. Disponible en: http://msdn2.microsoft.com/es-es/library/fx6bk1f4(VS.80).aspx.

27 Colaboradores de Wikipedia. SharpDevelop 2007. Disponible en: http://es.wikipedia.org/w/index.php?title=SharpDevelop&oldid=9572226.

28 Colaboradores de Wikipedia. Framework, 2007. Disponible en: http://es.wikipedia.org/w/index.php?title=Framework&oldid=8678758

29 Ídem [20].

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

31 (CIC), C. I. C. ¿Qué es HL7? , 2007. Disponible en: http://www.hl7spain.org/VerPagina.asp?IDPage=0

Bibliografía

(CIC), C. I. C. ¿Qué es HL7? , 2007. [Disponible en: http://www.hl7spain.org/VerPagina.asp?IDPage=0

(UAG), U. A. D. G. Coordinación de Informática Médica 2003. [Disponible en: http://cim.uag.mx/historia.html

AUTORES, C. D. CARE2X - El Proyecto, 2004. [Disponible en: http://www.care2x.org/

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

CÁRDENAS, M. A. C. Mono, lleva el .Net Framework a plataformas no-Windows., 2002. [Disponible en: http://www.mentores.net/articulos/Mono_Net_Framework_plataformas_noWindows.htm

CORPORATION, M. MSDN, 2007. [Disponible en: http://msdn2.microsoft.com/es-es/library/default.aspx

FOUNDATION, F. S. Acerca de los Estándares de Codificación de GNU Pascal, 2005. [Disponible en: http://www.gnu-pascal.de/h-gpcs-es.html

GARZÁS, J. Complejidad Ciclomática, o la métrica esencial para apreciar la calidad de un diseño 2006. [Disponible en: http://jgarzas.googlepages.com/complejidadcilomatica

GRACIA, J. Introducción al .NET Framework, 2004. [Disponible en: http://www.webestilo.com/aspnet/aspnet00.phtml

LEDO, A. D. R. Y. M. V. Informática en la salud pública cubana 2006. [Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-34662006000300015&lng=es&nrm=iso&tlng=es

LÓPEZ, D. B. L. G. SALUD, proposición de un diseño y premisas teóricas de una historia clínica computarizada para la atención hospitalaria., 2002. [Disponible en: http://www.cecam.sld.cu/pages/rcim/revista_3/articulos_html/articulo_boris.htm

PÉREZ, I. N. Introducción a JavaScript, 2006. [Disponible en: http://www.elcodigo.net/tutoriales/javascript/javascript1.html

S.L., L. Caracteristicas de ASP.NET, 2005. [Disponible en: http://www.adrformacion.com/cursos/aspnet/leccion1/tutorial1.html

SECO, J. A. G. El lenguaje de programación C#, 2001. [2007]. Disponible en: http://www.josanguapo.com/

ULTRASIST, I. E. S. A. ASP, 2004. [Disponible en: http://www.ultrasist.com.mx/tecnologias/asp.htm

VISUAL LIMES, S. L. FARHOS, 2005. [Disponible en: http://www.visual-limes.com/es/index.html?sec=pro&subsec=farhos&ap=quees

(CIC), C. I. C. ¿Qué es HL7? , 2007. [Disponible en: http://www.hl7spain.org/VerPagina.asp?IDPage=0

Anexos

En este tópico se mostrarán imágenes de las interfaces de usuario, cada imagen está vinculada con las clases interfaces descritas en el capítulo 2 epígrafe 2.8.4.

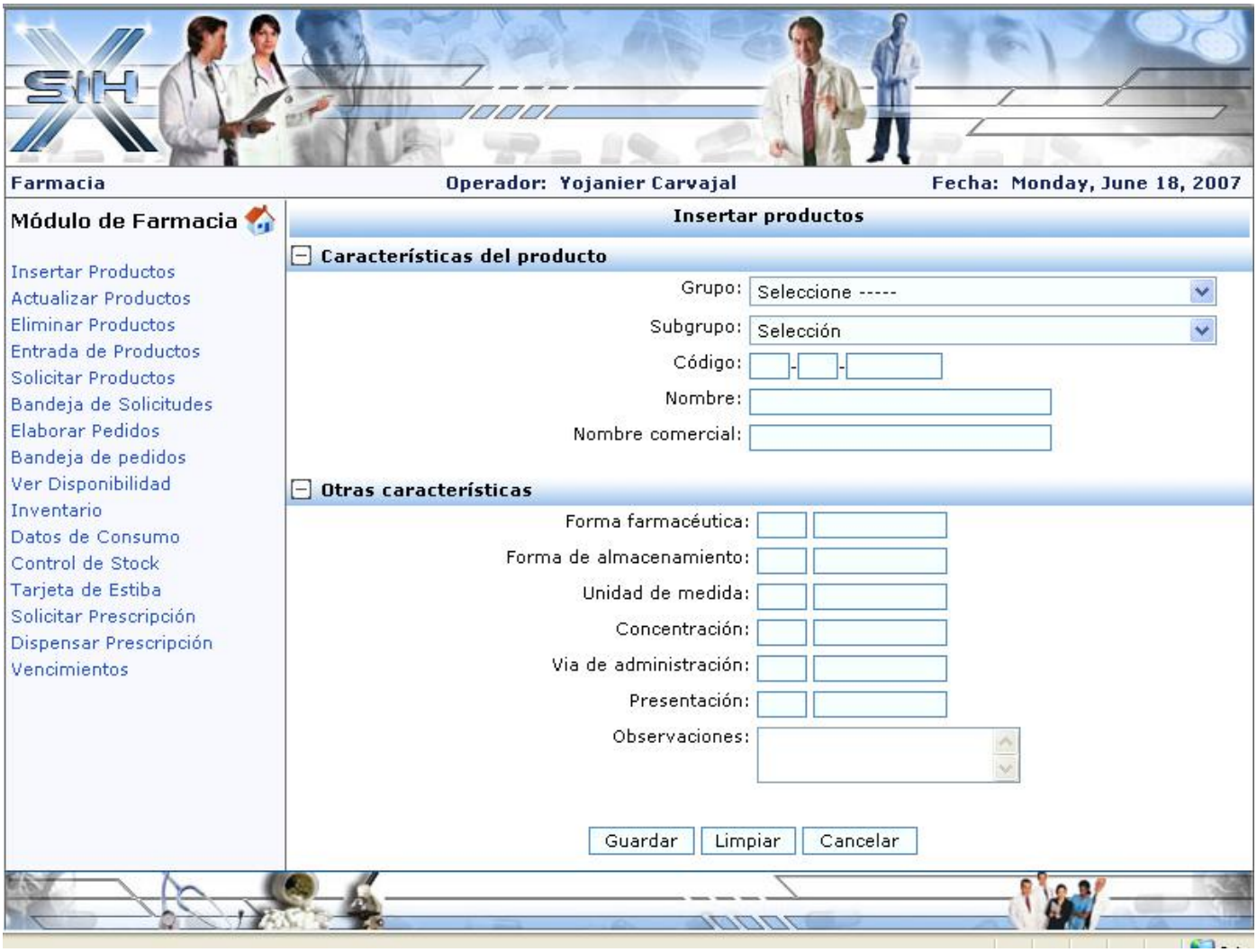

Figura 6 Página "Insertar Productos".

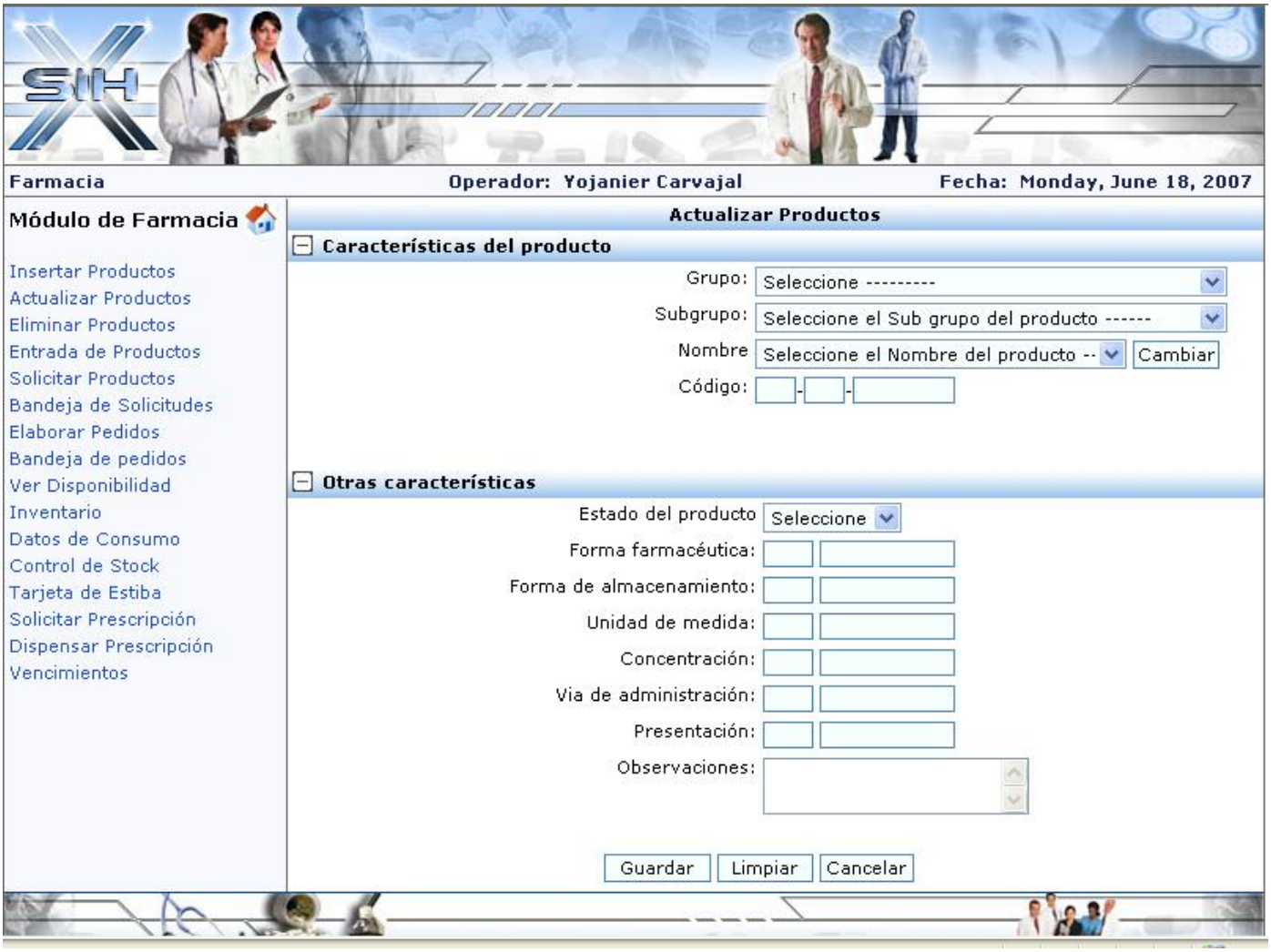

Figura 7 Página "Actualizar Productos".

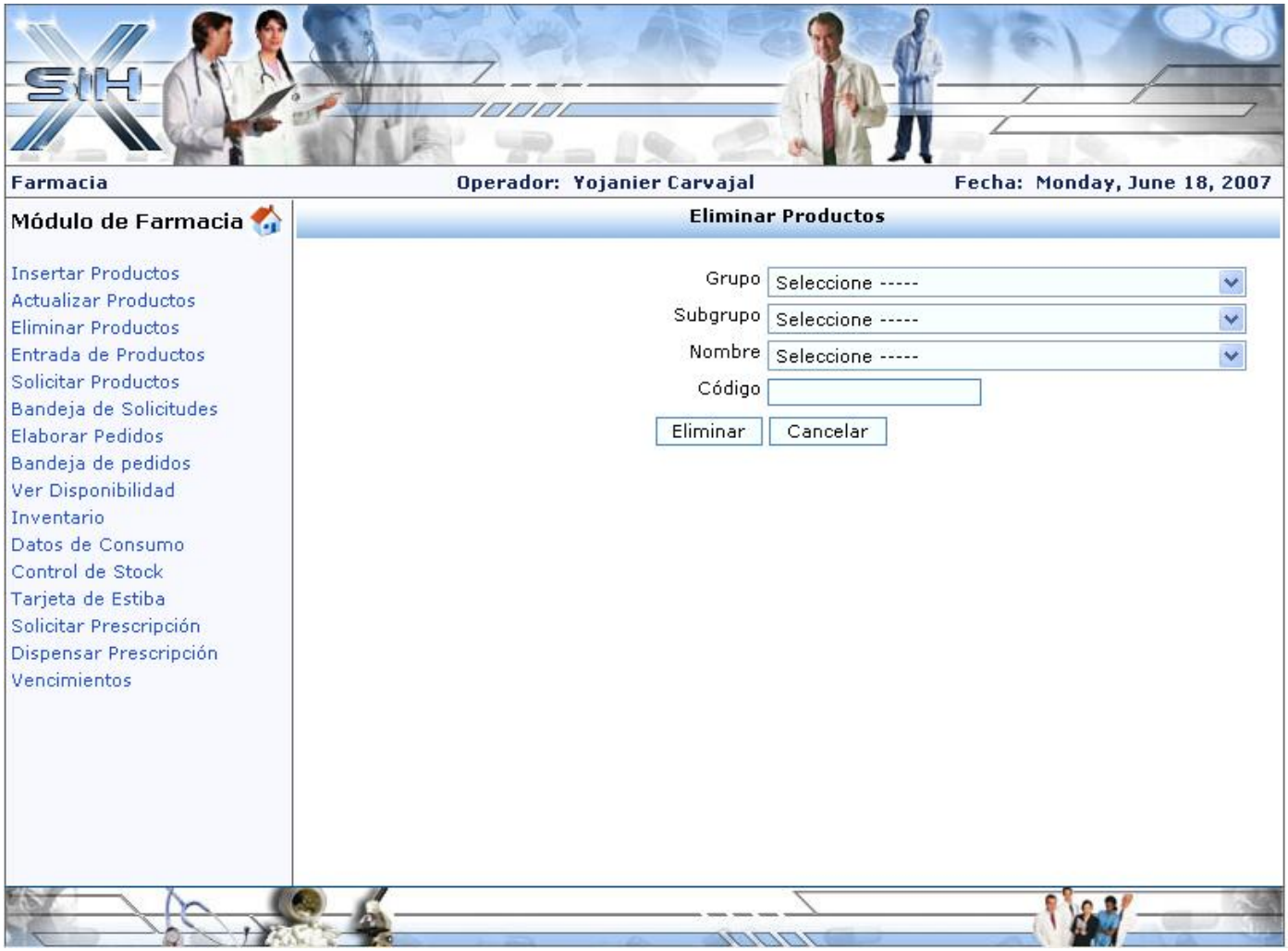

Figura 8 Página "Eliminar Productos".

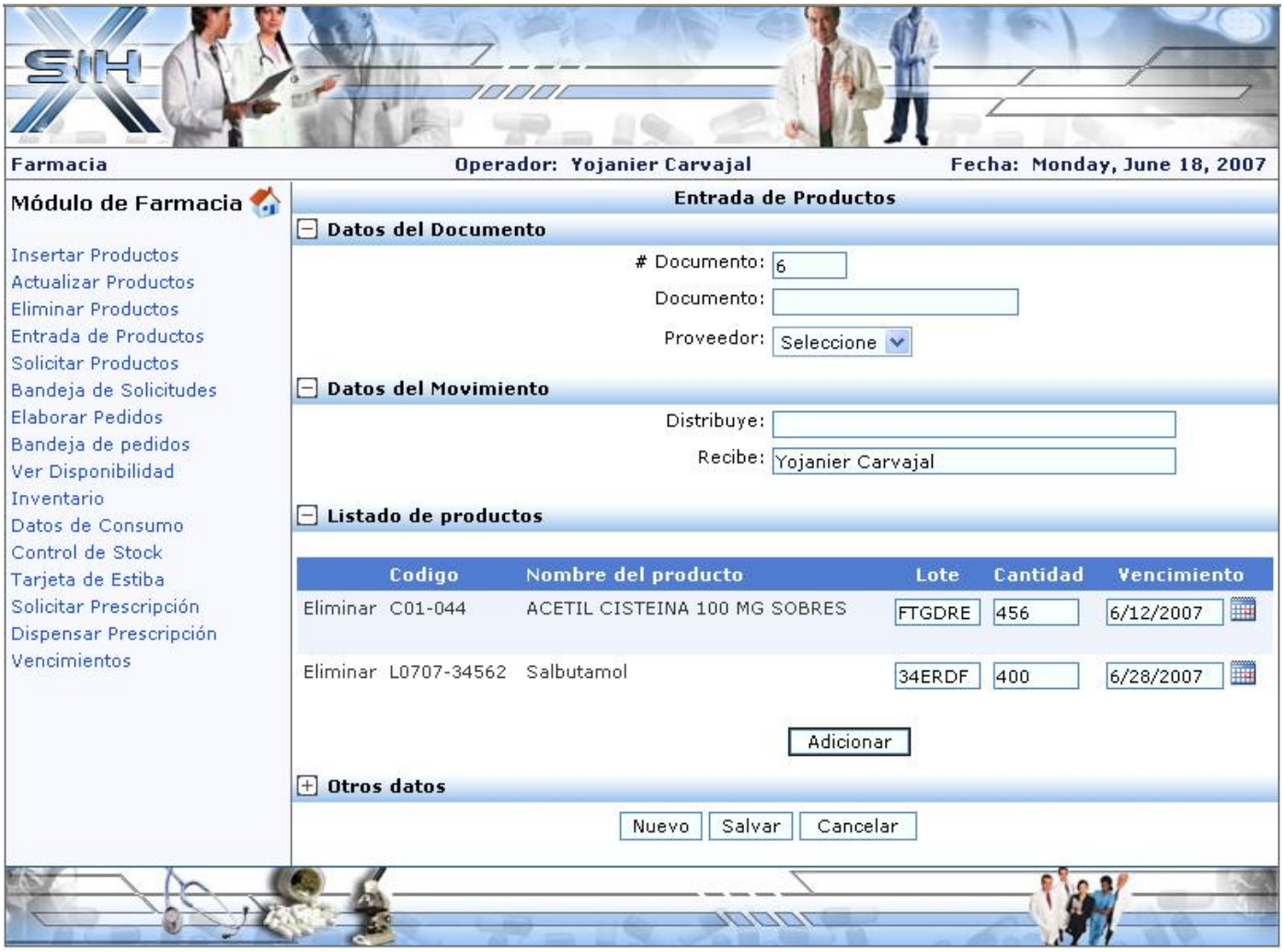

Figura 9 Página "Entrar Productos".

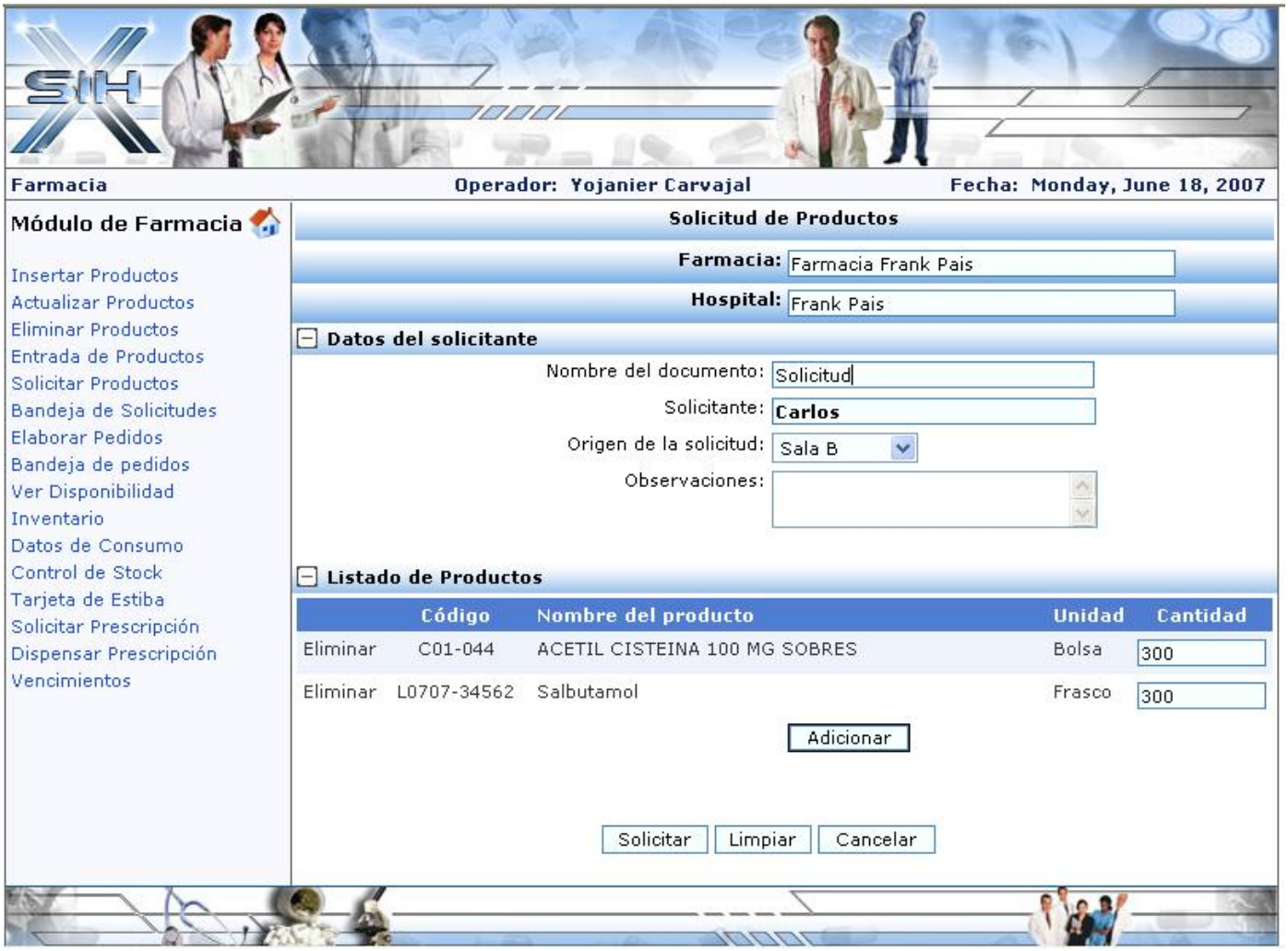

Figura 10 Página "Solicitar Productos".

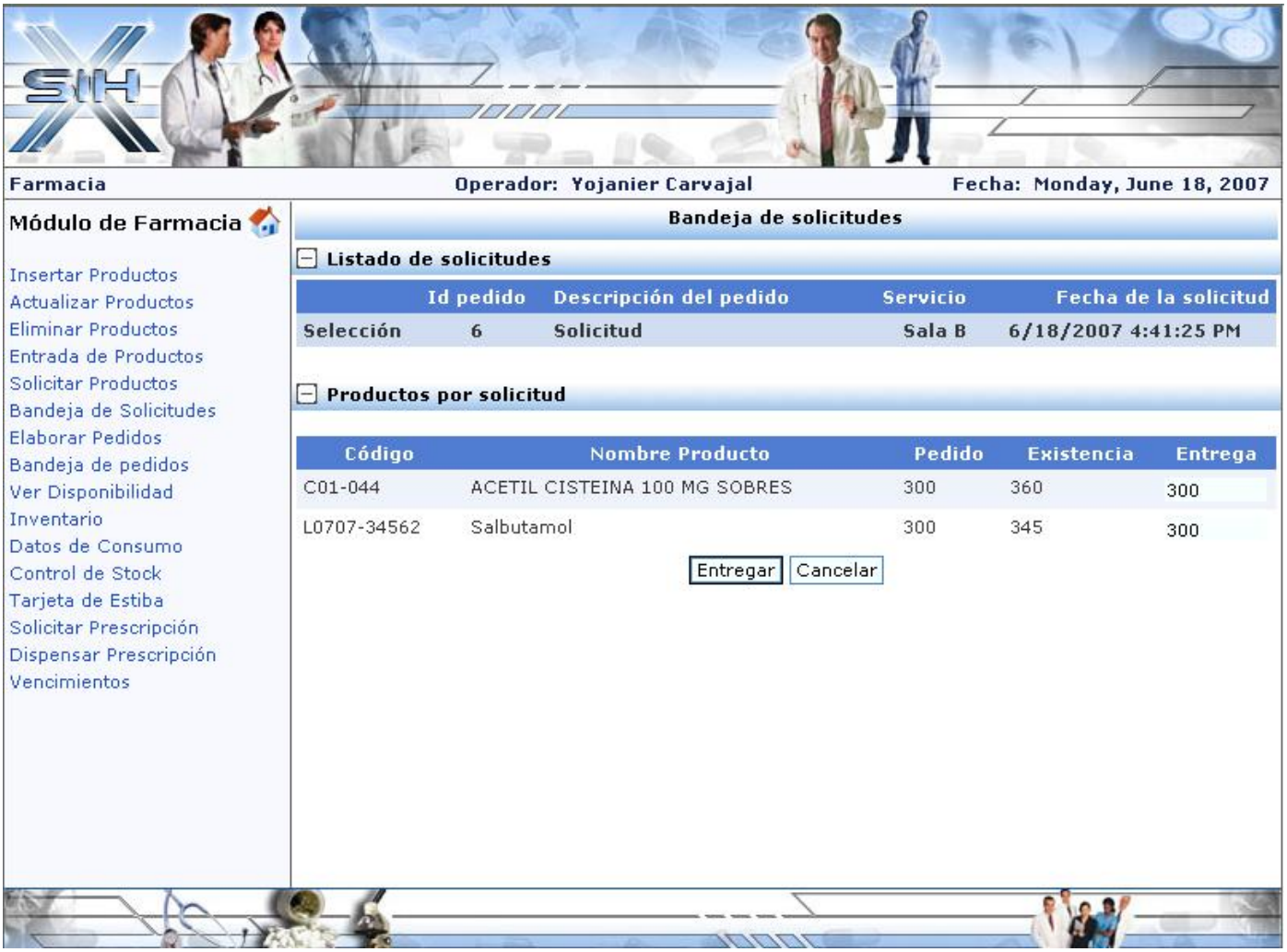

Figura 11 Página "Bandeja de Solicitudes".

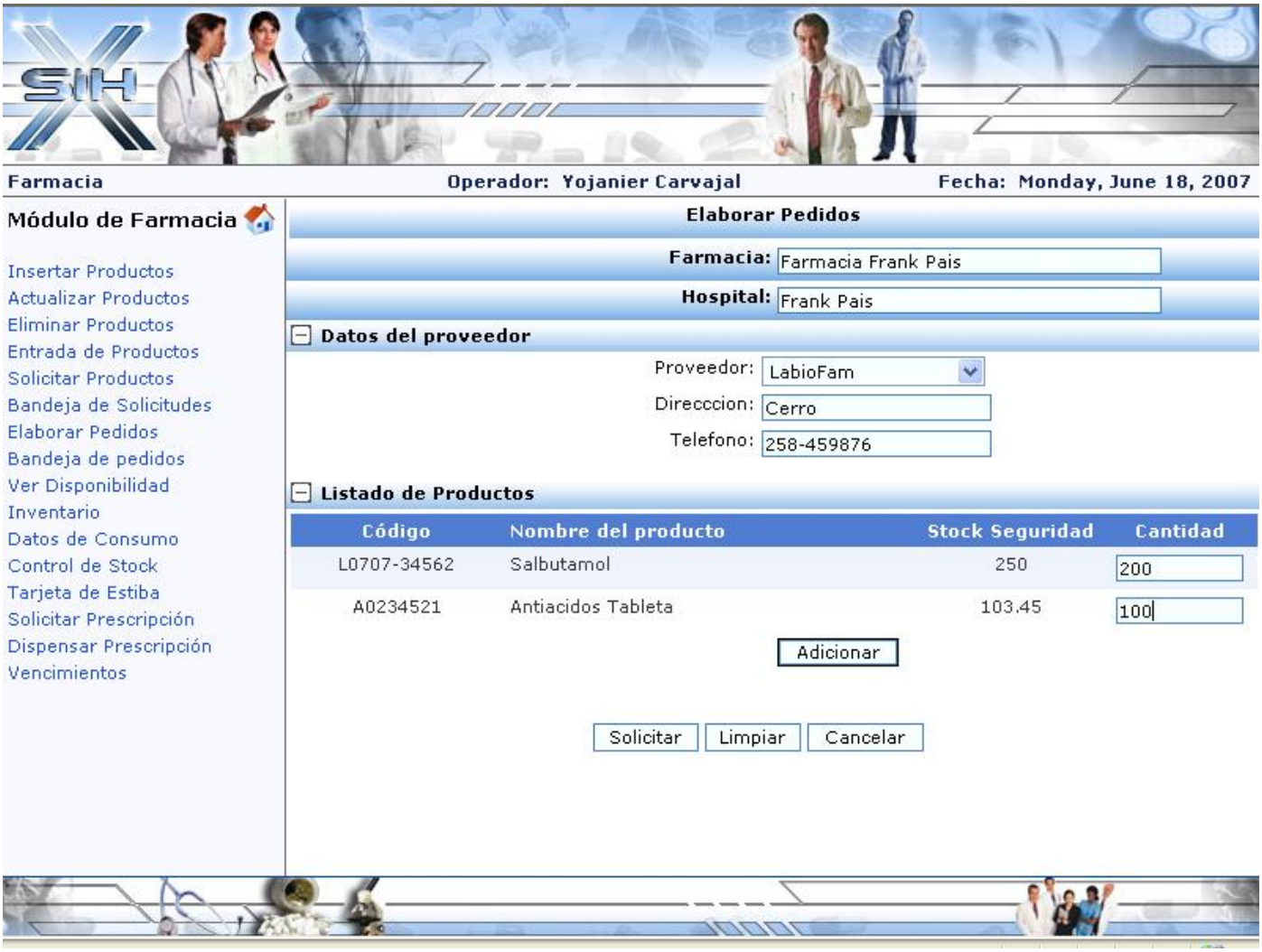

Figura 12 Página "Elaborar Pedidos".

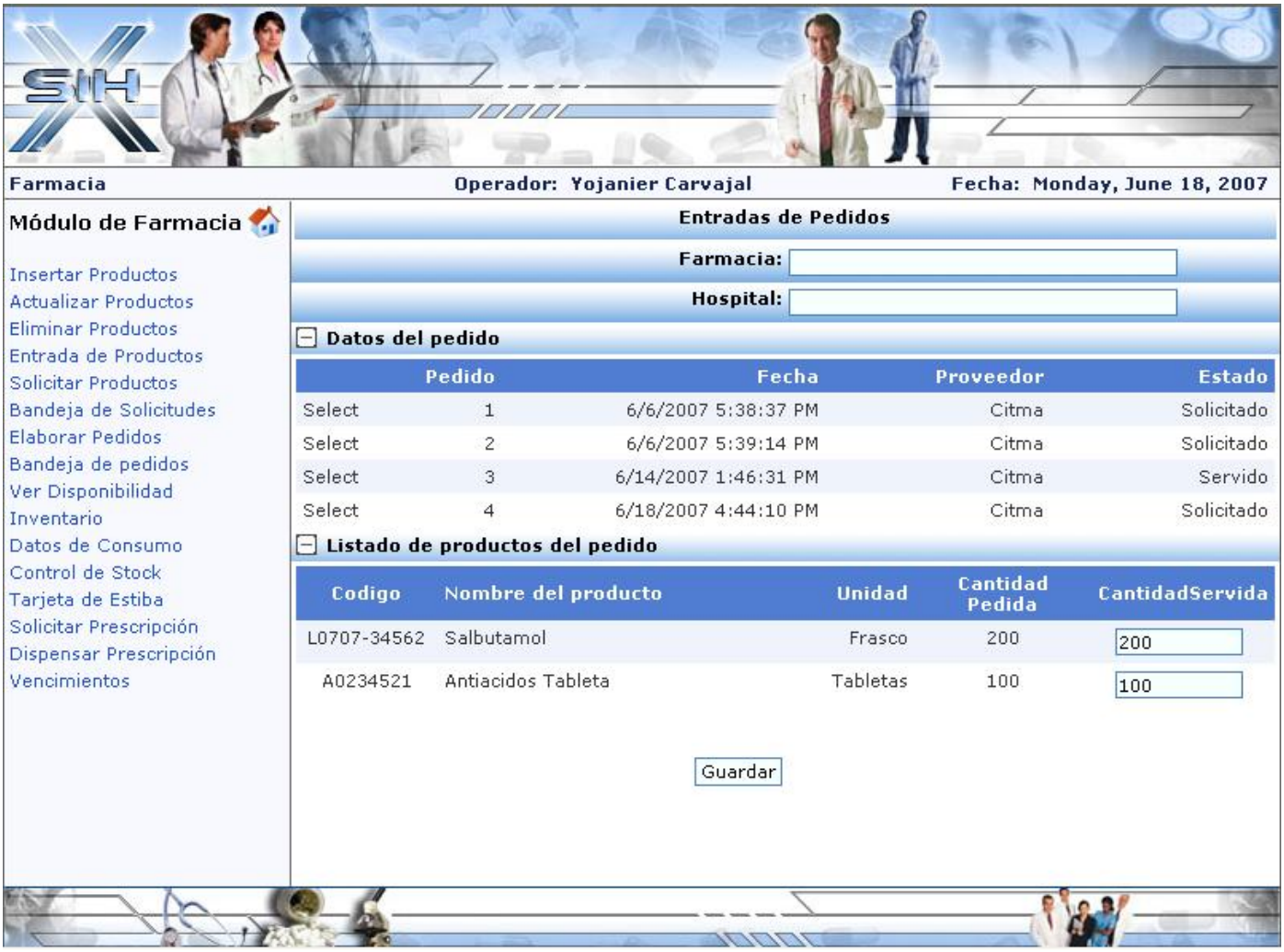

Figura 13 Página "Bandeja de Pedidos".

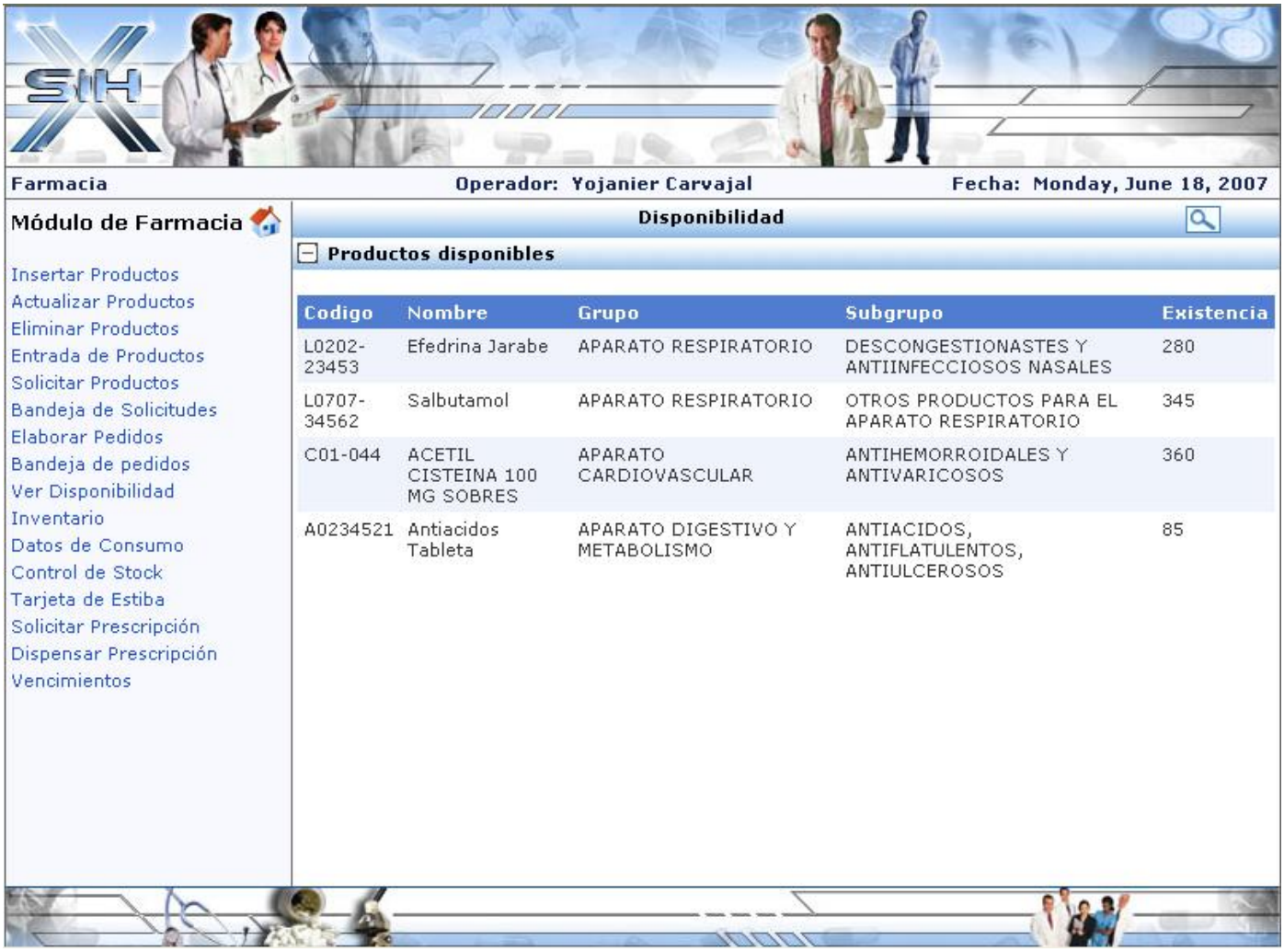

Figura 14 Página "Ver Disponibilidad".

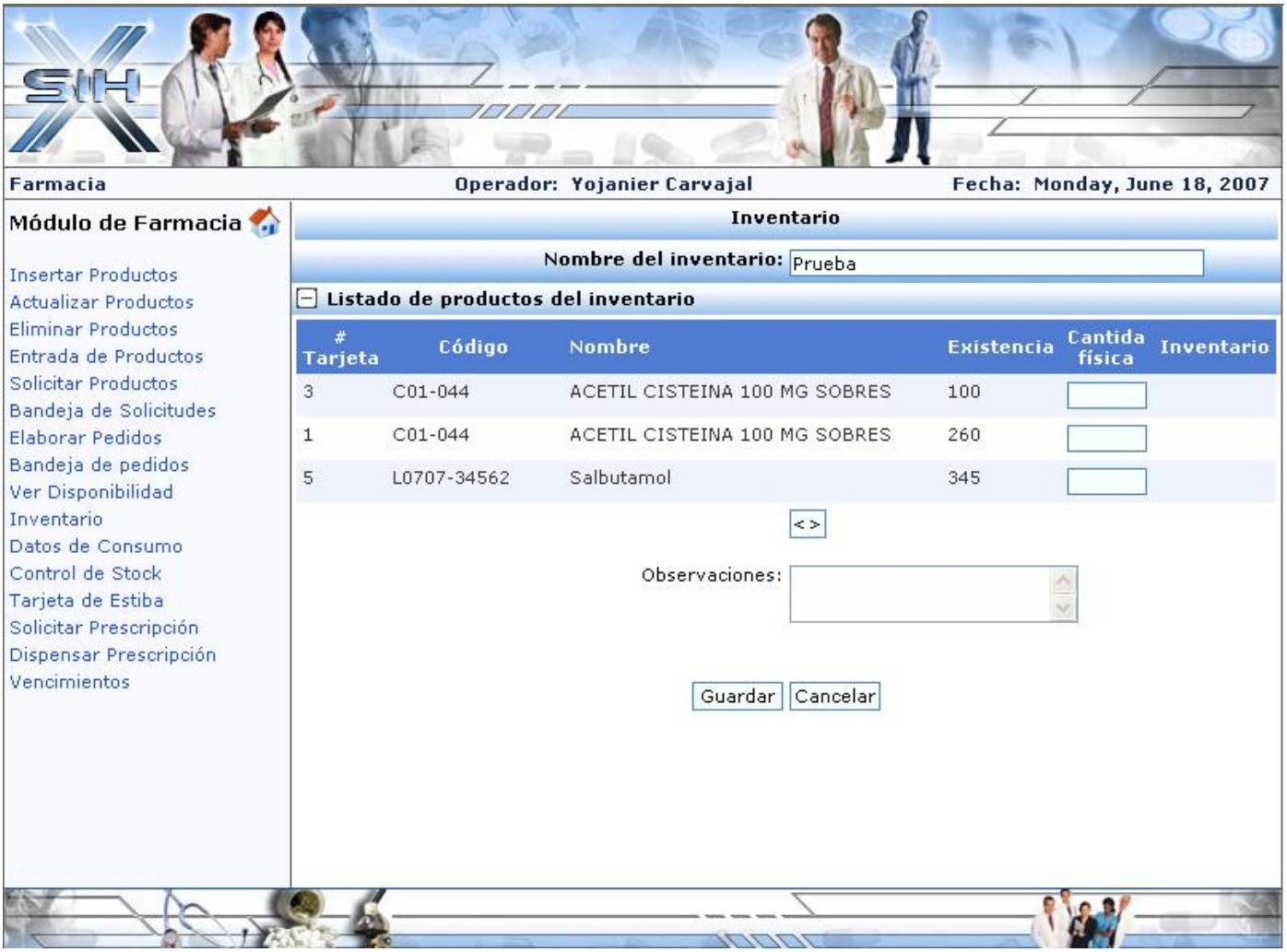

Figura 15 Página "Inventario".

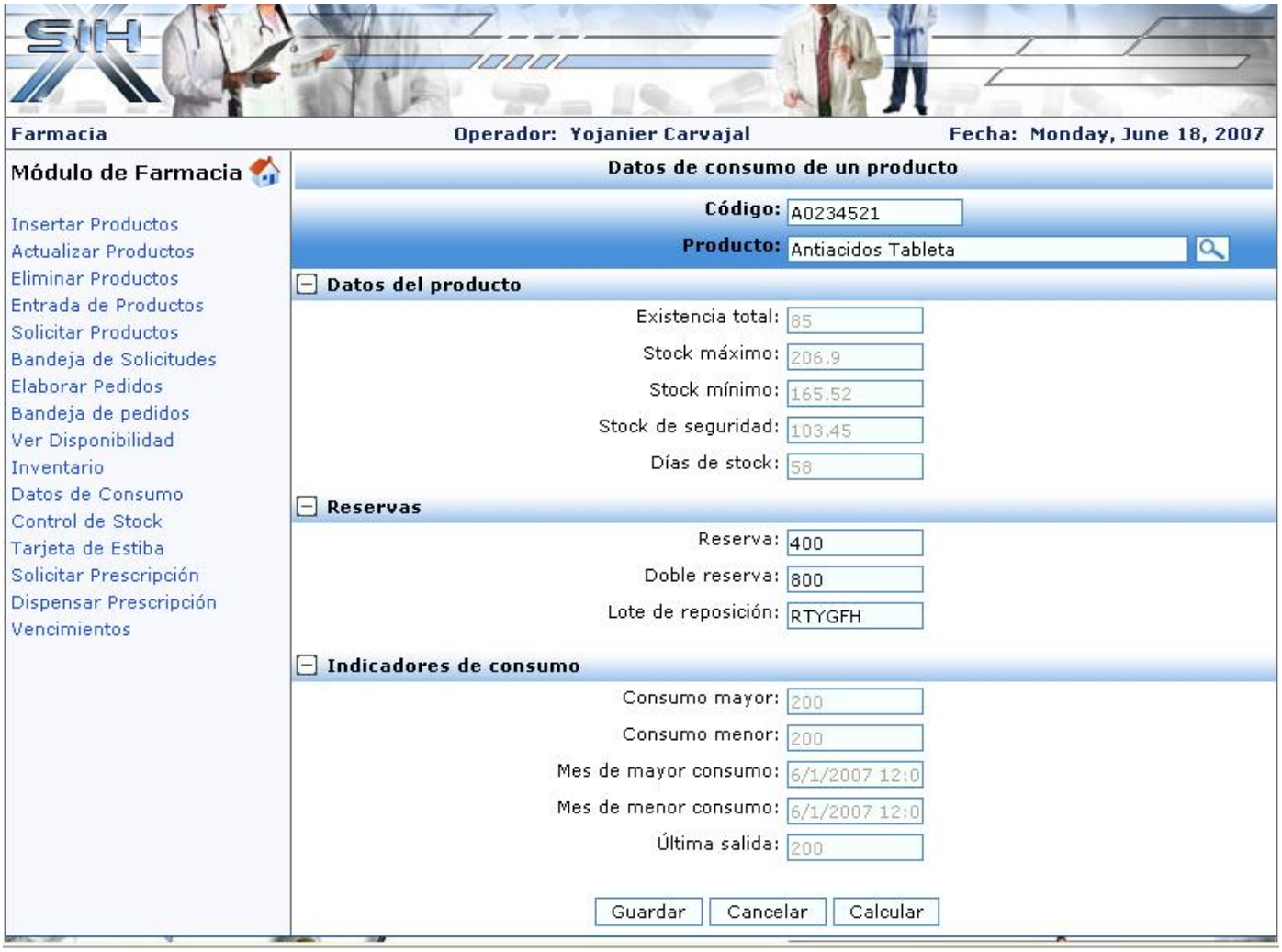

Figura 16 Página "Datos de Consumo".

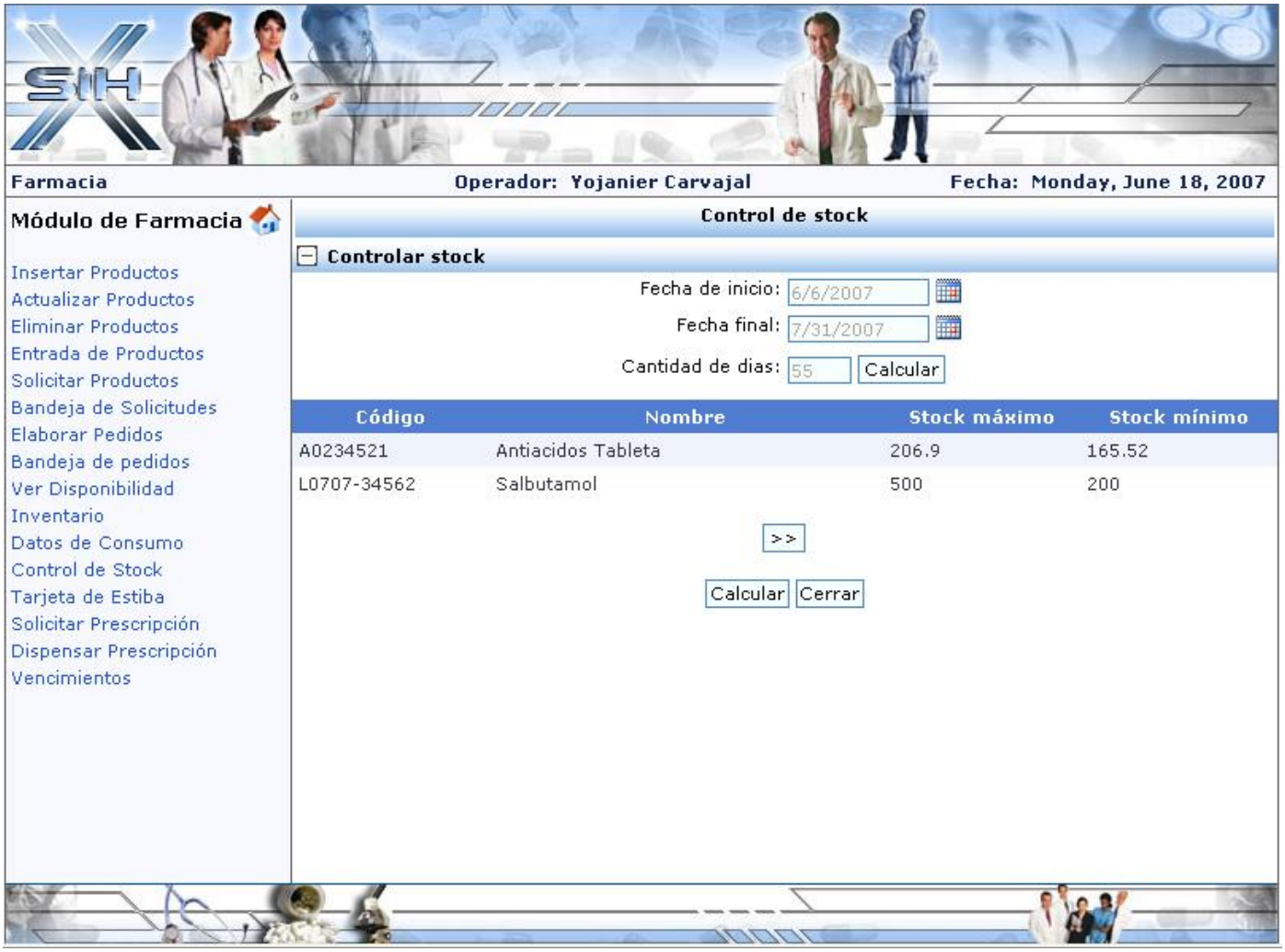

Figura 17 Página "Control de Stock".

Figura 18 Página "Tarjeta de Estiba".

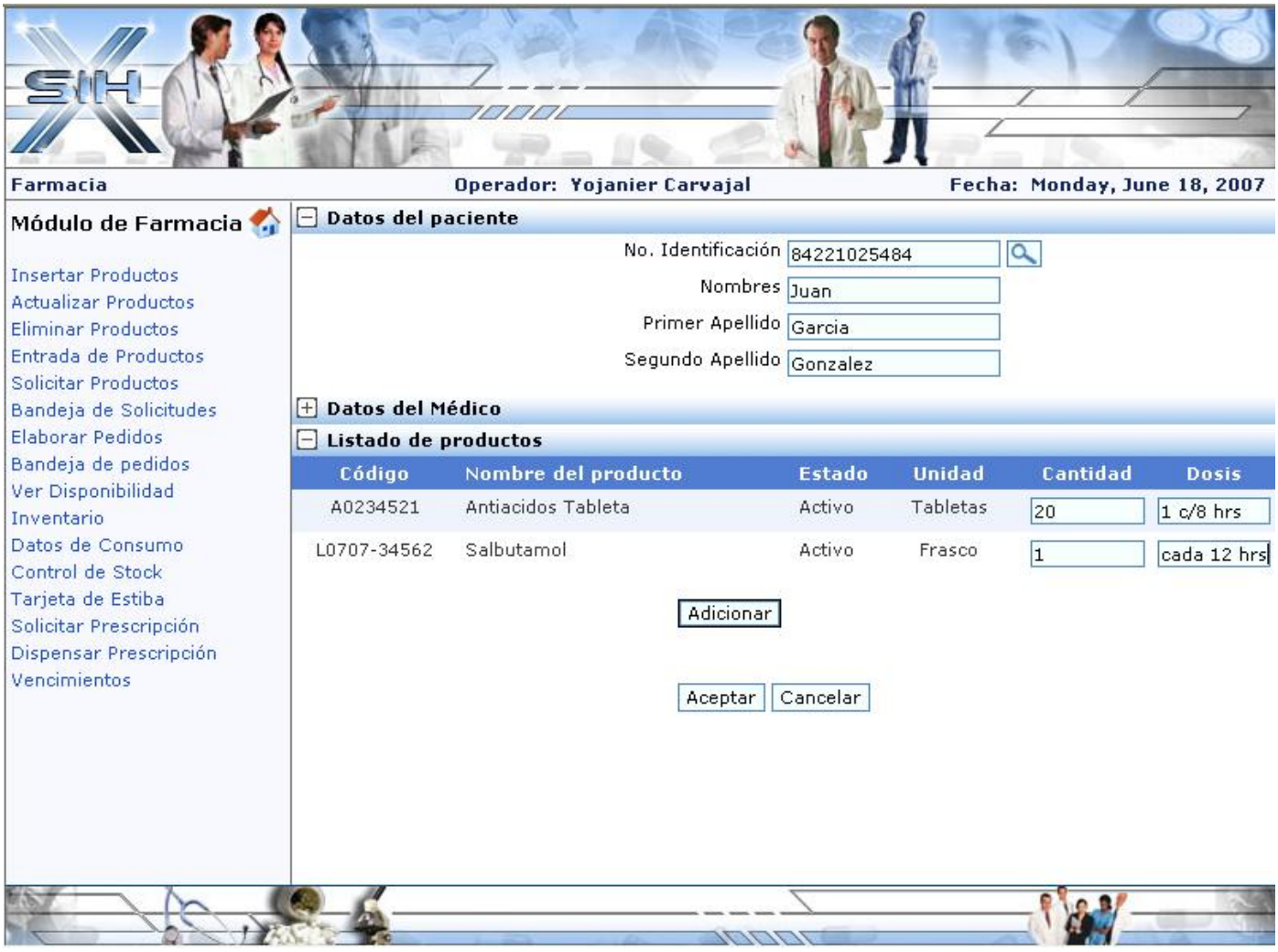

Figura 19 Página "Solicitar Prescripción".

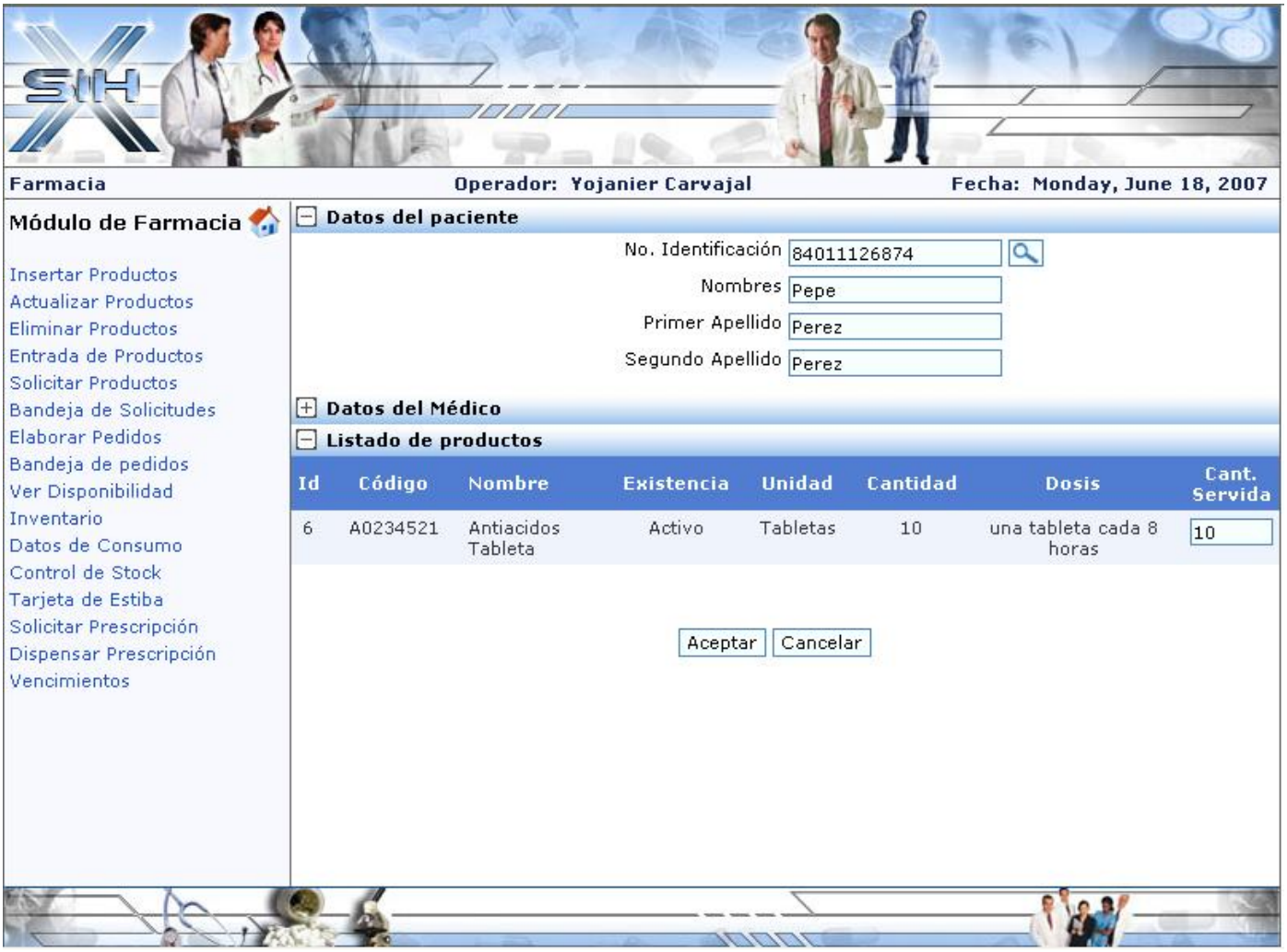

Figura 20 Página "Dispensar Prescripción".

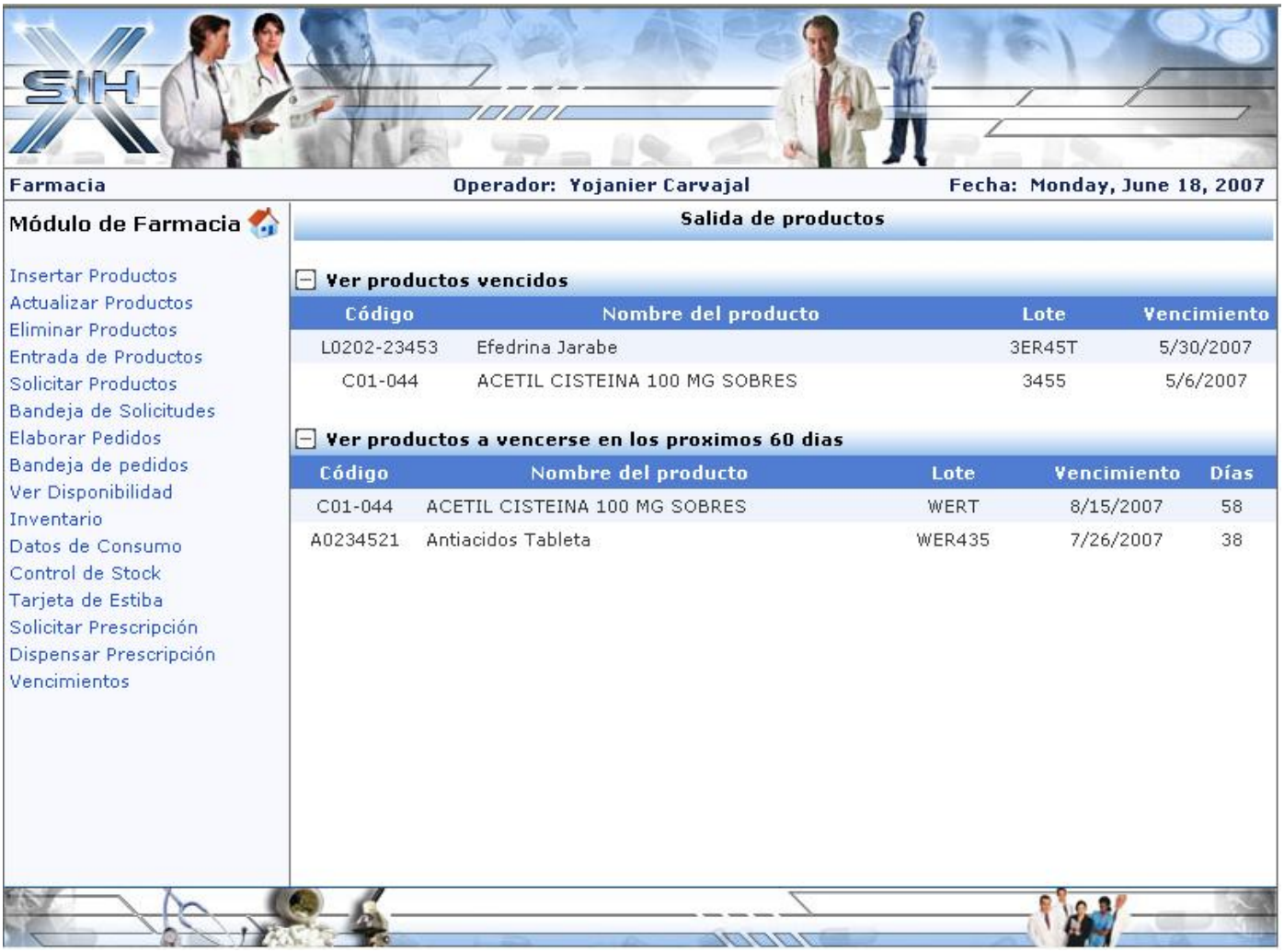

Figura 21 Página "Vencimientos".

Glosario de términos

Algoritmo: (en latín *algorithmus)* es un conjunto ordenado y finito de operaciones que permite hallar la solución de un problema. Un algoritmo es un sistema por el cual se llega a una o varias soluciones, teniendo en cuenta que debe ser definido, finito y eficiente. Este término no es exclusivo de las matemáticas, la informática o las ciencias de la computación ya que es muy usado en la genética.

ClickOnce: es una nueva tecnología de implementación que hace que la tarea de implementar una aplicación basada en Formularios Windows sea tan fácil como si fuera una aplicación Web.

Compilador JIT: (Just-In-Time) genera el código máquina real que se ejecuta en la plataforma que tenga la computadora.

Estándar de facto: es aquel patrón o norma que se caracteriza por no haber sido consensuada ni legitimada por un organismo de estandarización al efecto, es una norma generalmente aceptada y ampliamente utilizada por iniciativa propia de un gran número de interesados.

GNOME: (***G****NU* ***N****etwork* ***O****bject* ***M****odel* ***E****nvironment*) es un entorno de escritorio para sistemas operativos de tipo Unix bajo tecnología X Window. Se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.

GTK#: es un grupo importante de bibliotecas o rutinas para desarrollar interfaces gráficas de usuario (GUI) para principalmente los entornos gráficos GNOME, XFCE y ROX de sistemas Linux. GTK+ es la abreviatura de GIMP toolkit (conjunto de rutinas para GIMP). Es software libre (bajo la licencia LGPL), multiplataforma y parte importante del proyecto GNU.

HTML: es el acrónimo inglés de HyperText Markup Language. Es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web.

Internet: es un método de interconexión de redes de computadoras implementado en un conjunto de protocolos denominado TCP/IP y garantiza que redes físicas heterogéneas funcionen como una red

(lógica) única. De ahí que Internet se conozca comúnmente con el nombre de "red de redes", pero es importante destacar que Internet no es un nuevo tipo de red física, sino un método de interconexión.

ISAPI: (Interfaz de programación de aplicaciones de servidor para Internet) Es una interfaz para el desarrollo de aplicaciones de servidor Web, desarrollado por Process Software y Microsoft Corporation, que se utiliza en lugar de CGI.

MSIL: (acrónimo de Microsoft Intermediate Language) es un bytecode que la Tecnología .NET de Microsoft utiliza para lograr independencia de la plataforma y seguridad en ejecución. MSIL es un lenguaje de un nivel de abstracción mayor que el de la mayoría de los códigos máquina de las computadoras existentes, e incluye instrucciones que permiten trabajar directamente con objetos (crearlos, destruirlos, inicializarlos, llamar a métodos virtuales, entre otros), tablas y excepciones (lanzarlas, capturarlas y tratarlas).

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

Protocolo FTP: (*File Transfer Protocol*) es un protocolo de transferencia de ficheros entre sistemas conectados a una red TCP basado en la arquitectura cliente-servidor, de manera que desde un equipo cliente se puede conectar a un servidor para descargar ficheros desde él o para enviarle los archivos propios, independientemente del sistema operativo utilizado en cada equipo.

Protocolo HTTP: (*HyperText Transfer Protocol*) es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas Web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido. También sirve el protocolo para enviar información adicional en ambos sentidos, como formularios con campos de texto.

Protocolo HTTPS: es una versión más segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP.

Glosario de Términos.

SMTP: (*Simple Mail Transfer Protocol*) o protocolo simple de transferencia de correo electrónico. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras o distintos dispositivos (PDA's, teléfonos móviles, etc.).

Software: son todos los componentes intangibles de una computadora, es decir, el conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema.

Web: (diminutivo de *World Wide Web*) es un sistema de documentos de hipertexto enlazados y accesibles a través de Internet. Con un navegador Web, un usuario visualiza páginas Web que pueden contener texto, imágenes u otros contenidos multimedia, y navegar a través de ellas usando hiperenlaces.

WYSIWYG: (acrónimo de *What You See Is What You Get*) Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final, frecuentemente el resultado impreso.