Universidad de las Ciencias Informáticas

Facultad 7

Título: Implementación del Módulo Salud Ambiental del Sistema Control Sanitario Internacional

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autores: Runer Céspedes Aldana
Omar Rodríguez Boutorina

Tutor: Ing. Pastor López Gómez

Asesor: Lic. Luis Manuel Hernández Amarales

Ciudad de La Habana
Julio de 2008
"Año 50 de la Revolución"

"Nuestro conocimiento es una pequeña isla en el enorme océano del desconocimiento".

Isaac Bashevis Singer

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los 4 días del mes de Julio del año 2008.

_____ _____

Omar Rodríguez Boutorina
Autor

Runer Céspedes Aldana
Autor

_____

Ing. Pastor López Gómez
Tutor

DATOS DE CONTACTO

Ing. Pastor López Gómez

Graduado de Ingeniero en Ciencias Informáticas en el año 2007 en la Universidad de las Ciencias Informáticas (UCI). Actualmente se desempeña como profesor de Práctica Profesional I en la Facultad 7. Atiende Seguimiento al Graduado en la Dirección de Ingreso, Ubicación Laboral.

Correo Electrónico: plopezg@uci.cu

Lic. Luis Manuel Hernández Amarales

Licenciado en Ciencias Sociales. Actualmente se desempeña como profesor de Economía Política perteneciente al Departamento de Humanidades de la Facultad 7

Correo Electrónico: luisamarales@uci.cu

DEDICATORIA

De Runer:

A mis padres, por haber sido mi fuente de inspiración, por su a poyo incondicional, por servir de faro y guía en la vida, por ser un motivo de orgullo y ejemplo a seguir.

A nuestro eterno Comandante en Jefe por haber tenido la maravillosa idea de crear esta universidad en la que nos graduamos.

De Omar:

A mis padres, por ser ambos un ejemplo a seguir, por enseñarme el camino correcto y por demostrarme que son los mejores del mundo.

AGRADECIMIENTOS

De Omar:

A mis padres, por el apoyo y la confianza.

A mi hermana, por el corazón tan grande que tiene.

A Niover (El Chino), por sus consejos y su apoyo.

A mi amigo y compañero de tesis Runer, por ser el cerebro de esta operación.

A Betty, por su ayuda y dedicación.

A Jeney, por su cariño, por ser tan bonita y especial.

A Enrique (el gordito), por su gran aporte a la causa.

A mis amigos de la Universidad, por compartir conmigo 5 años que nunca olvidaré.

A mis amigos de Santa Cruz del Norte, porque somos como hermanos.

A todos los profesores que han contribuido en la formación de este joven revolucionario.

Al tutor, por su apoyo brindado.

Y a todo aquel que de una forma u otra ha contribuido con la realización de este sueño.

De Runer:

A mi madre, porque a pesar de estar muy lejos en estos momentos en el cumplimiento del deber, su ejemplo y sacrificio siempre estuvieron presentes.

A mi padre, por servir de guía para enfrentar los momentos más difíciles con absoluta serenidad.

A mi hermana, por mostrar su preocupación y su deseo de verme ingeniero, aunque tal vez no tenga conciencia de las dimensiones de tal empeño.

A todos mis familiares, en especial a mi tía Marilis, por ser mi segunda madre.

A todos los profesores que he tenido a lo largo de mi vida como estudiante, sin ellos no hubiese sido posible adquirir los conocimientos que hoy poseo.

A mi compañero de tesis, por su carisma, entusiasmo y dinamismo en la operación.

A mis amigos de la universidad, por ser como hermanos y compartir conmigo estos años inolvidables.

A Enrique, por contribuir desinteresadamente a mejorar la calidad de este trabajo.

A mi tutor, por el apoyo brindado.

A todos los que de una forma u otra aportaron su granito de arena en el desarrollo de este trabajo y en mi formación.

RESUMEN

El presente trabajo responde a la necesidad actual que tiene el Ministerio de Salud Pública de mejorar el control y la vigilancia sanitaria sobre los alimentos importados a Cuba. Actualmente, la gestión de esta no se realiza de forma eficiente. Para darle solución a la situación planteada se decidió implementar una aplicación Web que facilite la gestión de la información relacionada con la higiene de los alimentos de importación en Cuba.

Para desarrollar este sistema, se utilizó un conjunto de herramientas y técnicas basadas fundamentalmente en software libre, entre las cuales se destacan el Framework CodeIgniter, lenguaje PHP 5 y el gestor de base de datos MySQL 5.

La aplicación recoge los datos de los alimentos que Cuba importa en el momento que entran al territorio nacional a través de los puertos y aeropuertos. También registra la información referente a la calidad de dichos productos durante las numerosas Inspecciones Sanitarias que se les practican.

La puesta en marcha de esta aplicación facilitará la gestión de la información relacionada con la higiene de los alimentos importados por el país. Permitirá que la Unidad Nacional de Salud Ambiental cuente con información actualizada sobre el control y la vigilancia sanitaria de estos alimentos, lo que posibilitará prevenir y evitar la propagación en Cuba de enfermedades exóticas, emergentes y re-emergentes. Así como agilizar la toma de decisiones y la adopción de las medidas necesarias a los distintos niveles del Sistema Nacional de Salud.

PALABRAS CLAVES: Control Sanitario Internacional, Salud Ambiental, Higiene de los Alimentos, Informatización, Implementación, CodeIgniter.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El Ministerio de Salud Pública (MINSAP) es el Organismo rector del Sistema Nacional de Salud, encargado de dirigir, ejecutar y controlar la política del Estado y del Gobierno en cuanto a la Salud Pública. Algunas de las funciones rectoras del MINSAP son ejercer el control y la vigilancia sanitaria de todos los productos que pueden tener influencia sobre la salud humana, normar las condiciones higiénicas de dichos productos en aquellos aspectos que puedan resultar agresivos a la salud del hombre, así como controlar el cumplimiento de las normas sanitarias a través de la Inspección Sanitaria Estatal (ISE).

Para dar cumplimiento a estas funciones se establece el programa de Control Sanitario Internacional, el cual recoge una serie de regulaciones sanitarias internacionales que aseguran el máximo de seguridad contra la difusión internacional de enfermedades. Este programa está dividido en tres subprogramas fundamentales: Vectores, Epidemiología y Salud Ambiental.

Salud Ambiental es la rama encaminada a estudiar cómo el medio ambiente afecta la salud del hombre. Dentro de esta rama se tienen en cuenta 5 factores fundamentales: el estado de las aguas potables, los residuales líquidos y sólidos, la Salud Escolar, la Salud Ocupacional y la higiene de los alimentos.

Este último factor es de vital importancia para el país teniendo en cuenta que hoy en día el comercio internacional de alimentos ha alcanzado elevados niveles, y se han producido cambios en el sistema de vida y en los hábitos alimentarios. Lo cual ha provocado un aumento en la difusión internacional de Enfermedades Transmitidas por Alimentos (ETA), convirtiéndose en una causa importante de mortalidad a nivel mundial.

En los últimos años se han producido varios brotes extremadamente graves de ETA en prácticamente todos los continentes. La región latinoamericana experimentó al menos 6.000 brotes de diversos tipos de enfermedades de origen alimentario entre 1993 y 2002, según las cifras ofrecidas por la Organización de las Naciones Unidas para la Agricultura y la Alimentación (FAO) y la Organización Mundial de la Salud (OMS).

Han sido descritos alrededor de 250 agentes causantes de ETA, entre los que se incluyen bacterias, virus, hongos, parásitos, priones, toxinas y metales. Constituyen unos de los problemas de salud más extendido en el mundo, por tal motivo el control y la vigilancia epidemiológica de los alimentos de importación en Cuba es vital para aplicar medidas preventivas oportunas que permitan el control y prevención de dichas enfermedades. En el país durante el año 2000 se notificaron 306 brotes producidos por agua y alimentos, mientras que en el 2003 la cifra fue de 257 brotes.

En Cuba para garantizar el suministro de productos inocuos, nutritivos y/o seguros a la población se han establecido acciones regulatorias en concordancia con las legislaciones internacionales. Por lo que cada producto que entra al país debe ser evaluado atendiendo a sus características físico-químicas, microbiológicas, toxicológicas y nutricionales. Aspecto al que se le da seguimiento a través de la vigilancia sanitaria durante las ISE.

El control de la higiene de los alimentos de importación demanda de información diaria desde los puertos y las Unidades Municipales de Higiene y Epidemiología (UMHE) de todo el país, atravesando el nivel provincial hasta llegar a la Unidad Nacional de Salud Ambiental (UNSA) ubicada en el MINSAP.

Actualmente la información relacionada con la higiene de los productos de importación es manejada individualmente por las áreas de salud, cuando debería ser manejada de forma centralizada a nivel nacional, lo que trae consigo que se afecte el control y la correcta gestión de dicha información por parte de los niveles superiores. Cuando una carga de un producto determinado llega al país, se recogen de forma manual un gran volumen de datos, lo que dificulta el manejo de la información y atenta contra la integridad de estos.

Los inspectores sanitarios se deben comunicar con el Instituto de Nutrición a través de un teléfono para conocer las especificaciones de los productos registrados. Debido a que la aplicación existente en dicha institución no cuenta con las funcionalidades necesarias para que estos puedan acceder a la información. Durante las inspecciones sanitarias, la información de los conflictos o incidencias ocurridas es recogida y transmitida a los niveles superiores por vía telefónica, todo lo cual atenta contra la disponibilidad de la información y afecta la toma rápida de decisiones.

Por lo tanto se ha identificado como **problema científico** a resolver el siguiente: ¿Cómo facilitar la gestión de la información relacionada con la higiene de los alimentos de importación en Cuba?

Para dar solución a la problemática anterior el **objeto de estudio** se enfoca hacia el proceso de gestión de la información referente al Control Sanitario Internacional en el país.

El **campo de acción** está enmarcado en el proceso de gestión de la información referente a la higiene de los alimentos de importación en Cuba.

Para materializar este trabajo se trazó como **objetivo general** implementar una aplicación Web que facilite la gestión de la información relacionada con la higiene de los alimentos de importación en Cuba.

Con la finalidad de darle cumplimiento al objetivo trazado se definieron las siguientes **tareas**:

- ✓ Valorar las tecnologías a utilizar para el desarrollo de la aplicación.
- ✓ Elaborar el Modelo de Implementación.
- ✓ Elaborar los artefactos necesarios que describan la Base de Datos.
- ✓ Diseñar la Base de Datos.
- ✓ Realizar la implementación del módulo utilizando los patrones de diseño establecidos.
- ✓ Integrar la aplicación con los componentes definidos del Sistema de Información para la Salud (SISalud).

Del trabajo se esperan los siguientes **aportes prácticos:**

- ✓ Que facilite la gestión de la información en lo que respecta a salud ambiental.
- ✓ Que contribuya al mejoramiento de la salud ambiental en nuestro país.
- ✓ Que permita prevenir, detectar la introducción y evitar la propagación en Cuba de enfermedades exóticas, emergentes y re-emergentes y adoptar las medidas necesarias con la retroalimentación adecuada a los distintos niveles del Sistema Nacional de Salud.

El contenido del presente trabajo está estructurado en tres capítulos distribuidos de la siguiente manera:

- ✓ **Capítulo 1 "Fundamentación Teórica".** Se valoran las tendencias y tecnologías actuales. Se realiza un estudio crítico y valorativo de las técnicas de programación, plataforma y librerías usadas para el desarrollo del módulo.
- ✓ **Capítulo 2 "Elementos de arquitectura".** Se presentan los elementos de arquitectura utilizados para llevar a cabo el desarrollo de la aplicación, donde se destacan los requisitos no funcionales del sistema propuesto, los patrones arquitectónicos presentes, así como el diagrama de componentes y la vista de despliegue.
- ✓ **Capítulo 3 "Descripción y análisis de la solución propuesta".** Se hace un análisis de los componentes o módulos ya existentes que fueron definidos para ser rehusados, la descripción de las tablas de la base de datos, así como la descripción de las clases utilizadas en la implementación de la aplicación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo, como punto de partida se abordan conceptos fundamentales relacionados con los términos web y aplicaciones web. Además se valoran las tendencias y tecnologías actuales así como las plataformas de desarrollo que las soportan. Estos elementos aportan criterios comparativos que permiten realizar un estudio crítico y valorativo de las técnicas de programación, plataforma y librerías usadas para el desarrollo del módulo.

1.1 Aplicaciones web

La World Wide Web consiste en ofrecer una interfaz simple y consistente para acceder a la gran cantidad de recursos disponibles en Internet. Es la forma más moderna de ofrecer y consumir información, además constituye el medio más potente para la difusión de la misma. La información se ofrece en forma de páginas electrónicas.

La Web introduce un concepto fundamental: la posibilidad de lectura universal, lo cual consiste en que una vez que la información esté disponible, se pueda acceder a ella desde cualquier ordenador, desde cualquier país, por cualquier persona autorizada, usando un único y simple programa (un navegador).

Internet y la Web han influido enormemente tanto en el mundo de la informática como en la sociedad en general. La Web en poco menos de 10 años ha transformado los sistemas informáticos, ha roto las barreras físicas (debido a la distancia), económicas y lógicas (debido al empleo de distintos sistemas operativos y protocolos.) y ha abierto todo un abanico de nuevas posibilidades. Una de las áreas que más expansión ésta teniendo en la Web en los últimos años son las aplicaciones web.

Las aplicaciones web permiten la generación automática de contenido, la creación de páginas personalizadas según el perfil del usuario o el desarrollo del comercio electrónico. Además, una aplicación web permite interactuar con los sistemas informáticos de gestión de una empresa, como puede ser gestión de clientes, contabilidad o inventario, a través de una página web. Las aplicaciones web se encuadran dentro de las arquitecturas cliente/servidor; un ordenador solicita servicios (el cliente) y otro está a la espera de recibir solicitudes y las responde (el servidor). (1)

La idea fundamental es que los navegadores (browsers) presentan documentos escritos en HTML que han obtenido de un servidor web. Estos documentos HTML habitualmente presentan información de forma estática, sin más posibilidad de interacción con ellos. (2)

El modo de crear los documentos HTML ha variado a lo largo de la corta vida de las tecnologías Web pasando desde las primeras páginas escritas en HTML almacenadas en un fichero en el servidor Web hasta aquellas que se generan al vuelo como respuesta a una acción del cliente y cuyo contenido varía según las circunstancias. (3)

Además, el modo de generar páginas dinámicas ha evolucionado, desde la utilización del CGI, Common Gateway Interface, hasta los servlets pasando por tecnologías tipo JavaServer Pages. Todas estas tecnologías se incluyen dentro de aquellas conocidas como Server Side, ya que se ejecutan en el servidor web. (4)

Otro aspecto que completa el panorama son las inclusiones del lado del cliente, Client Side, que se refieren a las posibilidades de que las páginas lleven incrustado código que se ejecuta en el cliente, como por ejemplo JavaScript y programas Java.

1.2 Tendencias y tecnologías actuales

1.2.1 Técnicas de programación

Programación Orientada a Objetos

La Programación Orientada a Objetos (POO u OOP según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo, y encapsulamiento. Su uso se popularizó a principios de la década de 1990. Actualmente varios lenguajes de programación soportan la orientación a objetos. (5)

Programación orientada a componentes

Variante natural de la programación orientada a objetos para los sistemas abiertos. Presenta como nuevos elementos: los aspectos composicionales distintos a los aspectos computacionales. Los objetos no son unidades de composición independientes de las aplicaciones. Las interfaces de los objetos son de muy bajo nivel para la reutilización. (6)
El principal objetivo es construir un mercado global de componentes cuyos usuarios son los propios desarrolladores de aplicaciones que necesitan reutilizar componentes ya hechos y probados para construir sus aplicaciones de forma más rápida y robusta o que quieren añadir funcionalidad dependiente de terceros.

Programación orientada a servicios

La programación orientada a servicios es un complemento de la programación orientada a objetos e implementa mejoras sustanciales sobre los objetos, producto de la experiencia acumulada en la última década, sobre todo en las áreas de la computación distribuida, instalación de una solución e interoperabilidad entre sistemas heterogéneos.

1.2.2 Lenguajes de programación

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un lenguaje informático. Aunque muchas veces se usa lenguaje de programación y lenguaje informático como si fuesen sinónimos, no tiene por qué ser así, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más.

Un lenguaje de programación permite a un programador especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural, tal como sucede con el lenguaje léxico. Un programa escrito en un lenguaje de programación necesita pasar por un proceso de compilación, es decir, ser traducido al lenguaje de máquina, o ser interpretado para que pueda ser ejecutado por el ordenador. (7)

Lenguajes del lado del cliente

HTML

HTML es el acrónimo inglés de HyperText Markup Language, que se traduce al español como Lenguaje de Marcas Hipertextuales. Es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores como Internet Explorer, Opera, Firefox, Netscape o Safari, el HTML se ha convertido en uno de los formatos más populares y fáciles de aprender que existen para la elaboración de documentos para la web.

El lenguaje HTML puede ser editado con cualquier editor de textos básico, como puede ser Gedit, el Bloc de Notas de Windows, o cualquier otro editor que admita texto sin formato como GNU Emacs, Microsoft Wordpad, TextPad y Vim. HTML utiliza etiquetas o marcas, que consisten en breves instrucciones de comienzo y final, mediante las cuales se determinan la forma en la que debe aparecer en su navegador el texto, así como también las imágenes y los demás elementos. (8)

JavaScript

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

Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas Web. El cual se ejecuta del lado del cliente, al mismo tiempo que las sentencias van descargándose junto con el código HTML. (9)

Lenguajes del lado del servidor

Active Server Page (ASP)

Active Server Pages (ASP) es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS). La tecnología ASP está estrechamente relacionada con el modelo tecnológico de su fabricante. Intenta ser solución para un modelo de programación rápida ya que programar en ASP es como programar en Visual Basic, por supuesto con muchas limitaciones ya que es una plataforma que no se ha desarrollado como lo esperaba Microsoft. (10)

Lo interesante de este modelo tecnológico es poder utilizar diversos componentes ya desarrollados como algunos controles ActiveX. Uno de los problemas que han hecho evolucionar esta tecnología es el no disponer de información que oriente a quienes desean aprenderla.

Profesional Home Page Tools (PHP)

PHP es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es un acrónimo recursivo que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o, Personal Home Page Tools), y se trata de un lenguaje interpretado. Últimamente también puede ser utilizado para la creación de otro tipo de programas incluyendo aplicaciones con interfaz gráfica.

La similitud con los lenguajes más comunes de programación estructurada, como C y Perl, permite a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy suave. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones y prácticas. (11)

1.2.3 Gestores de base de datos

MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual.

MySQL es un gestor de base de datos sencillo de usar e increíblemente rápido. También es uno de los motores de base de datos más usados en Internet, la principal razón de esto es que es gratis para aplicaciones no comerciales. (12)

Las características principales de MySQL son:

- ✓ Es una base de datos relacional.
- ✓ Es de código abierto.
- ✓ Es una base de datos muy rápida, segura y fácil de usar.
- ✓ Existe una gran cantidad de software que la usa.

Microsoft SQL Server

Microsoft SQL Server es un sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje Transact-SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Entre sus características figuran: (13)

- ✓ Escalabilidad, estabilidad y seguridad.

- ✓ Soporta procedimientos almacenados.
- ✓ Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- ✓ Permite trabajar en modo cliente-servidor donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- ✓ Además permite administrar información de otros servidores de datos.

PostgreSQL

Es un servidor de base de datos objeto relacional, liberado bajo la licencia BSD. Como muchos otros proyectos de código abierto, su desarrollo no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales, las cuales trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). (14)

Algunas de sus principales características son:

- ✓ Alta concurrencia.
- ✓ Amplia variedad de tipos nativos.
- ✓ Claves ajenas.
- ✓ Disparadores.
- ✓ Vistas.
- ✓ Integridad transaccional.
- ✓ Herencia de tablas.
- ✓ Tipos de datos y operaciones geométricas.

1.2.4 Servidores Web

Un servidor web es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir hipertextos, páginas web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

Es un programa que se ejecuta continuamente en un ordenador (también se emplea el término para referirse al ordenador que lo ejecuta), manteniéndose a la espera de peticiones por parte de un cliente

(un navegador de Internet) que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error.

Internet Information Server

Internet Information Server (IIS) es un servidor web, que incluye los servicios de: HTTP, HTTPS, FTP, SMTP (correo saliente) y NNTP (grupos de noticias). Además es capaz de ejecutar varios motores de script como: ASP, PHP y Cold Fusion.

IIS es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS.

Este servicio convierte a un ordenador en un servidor de Internet o Intranet es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente (servidor web). (15)

Apache

El servidor HTTP Apache es un software libre, de código abierto para plataformas Unix, Windows, Macintosh y otras, que implementa el protocolo HTTP. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. (16)

Presenta, entre otras características, mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración. (17)

Apache tiene una amplia aceptación en la red: desde 1996, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años.

1.2.5 Metodologías de desarrollo

Proceso Unificado de desarrollo (RUP)

El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto al Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. (18)

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Posee tres características fundamentales, la primera de ellas es que su desarrollo es iterativo e incremental; teniendo un producto final al concluir cada ciclo. En cada uno de estos ciclos hace exigencia del uso de artefactos para lograr los hitos propuestos en cada una de las fases y obtener un incremento en el proceso de desarrollo. Por este motivo es considerada una de las metodologías más importante para alcanzar un grado de certificación en el desarrollo del software.

La segunda es que está guiado por los casos de uso. Un caso de uso será aquello que describe un fragmento de las funcionalidades del sistema que proporciona al usuario un resultado importante. Los casos de uso guían el diseño, construcción y prueba del sistema, esto significa que guían el proceso de desarrollo.

Por último, RUP está centrada en la arquitectura, lo que le permite a los desarrolladores una mayor visibilidad del sistema, pues la arquitectura es una vista del diseño completo del software con las características más importantes resaltadas, dejando a un lado los detalles.

Extreme Programing (XP)

La programación extrema o *eXtreme Programming* (XP) es un enfoque de la ingeniería de software formulado por Kent Beck. Es la más destacada de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más

realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlas de manera dinámica durante el ciclo de vida del software. (19)

Las características fundamentales del método son:

- ✓ **Desarrollo iterativo e incremental**: pequeñas mejoras, unas tras otras.
- ✓ **Pruebas unitarias continuas**, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- ✓ **Programación en parejas**: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
- ✓ Frecuente **integración del equipo de programación con el cliente** o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- ✓ **Corrección de todos los errores** antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- ✓ **Refactorización del código**, es decir, reescribir ciertas partes del código para aumentar su legibilidad y optimización, pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- ✓ **Propiedad del código compartida**: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- ✓ **Simplicidad** en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Mientras más simple es el sistema, menos tendrá que comunicar sobre este, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores. (20)

Microsoft Solution Framework (MSF)

Es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. Se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Esta metodología tiene las siguientes características:

- ✓ Adaptable, es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un lugar específico.
- ✓ Escalable, puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas o más.
- ✓ Flexible, es utilizada en el ambiente de desarrollo de cualquier cliente.
- ✓ Tecnología agnóstica, porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el Modelo de Aplicación (21)

1.2.6 Frameworks de desarrollo

En el desarrollo de software, un ***framework*** es considerado 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 interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Los *framework* 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.

.NET

.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de aplicaciones. Basado en esta plataforma, Microsoft intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el Sistema Operativo hasta las herramientas de mercado. (22)

Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones –o como la misma plataforma las denomina, soluciones– permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

J2EE

Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación (parte de la Plataforma Java) para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de n niveles distribuida, basándose ampliamente en componentes de software modulares que se ejecutan sobre un servidor de aplicaciones.

La plataforma Java EE está definida por una especificación, similar a otras especificaciones del Java Community Process. Es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE. (23)

MONO

Mono es el nombre de un proyecto de código abierto iniciado por la empresa Ximian y actualmente impulsado por Novell (tras su adquisición de Ximian para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET según lo especificado por el ECMA.

Mono posee importantes componentes útiles para desarrollar software:

- ✓ Una máquina virtual de lenguaje común de infraestructura (CLI) que contiene un cargador de clases, un compilador en tiempo de ejecución (JIT), y unas rutinas de recolección de memoria.
- ✓ Una biblioteca de clases que puede funcionar en cualquier lenguaje que funcione en el CLR (Common Language Runtime).

- ✓ Un sistema de objetos único, sistema de hilos, bibliotecas de clases y un sistema recolector de memoria, pueden ser compartidos por todos estos lenguajes.

Es un proyecto independiente de la plataforma. Actualmente Mono corre sobre Linux, FreeBSD, UNIX, Mac OS X, Solaris y plataformas Windows. (24)

1.2.7 Frameworks que usan el patrón arquitectónico Modelo-Vista-Controlador

En la actualidad existen numerosos **framewoks** que se apoyan en el uso de la estructura MVC ("Model-View-Controller") para desarrollar diversas aplicaciones web. A través de ellos se logra una división de las diferentes partes que conforman la aplicación, siendo el mantenimiento del código fuente su principal razón de ser.

A medida que se incrementan las funcionalidades de cualquier aplicación, la modificación del código existente se hace imprescindible y si no existe una clara división del sistema en partes con responsabilidades bien definidas, el código no solo se torna indescifrable sino en ocasiones impredecible debido a la mezcla de funcionalidades que pueden surgir. A continuación se exponen las características fundamentales de algunos de esos framewoks.

Ruby on Rails

Ruby on Rails, también conocido como **RoR** o **Rails** es un framework de código abierto escrito en el lenguaje de programación Ruby que permite el desarrollo ágil de aplicaciones web, siguiendo el paradigma de la arquitectura MVC. Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso.

CakePHP

Es otro framework de desarrollo de aplicaciones web, pero a diferencia del anterior está escrito en PHP, aunque fue creado sobre los conceptos de Ruby on Rails.

Al igual que Ruby On Rails, CakePHP facilita al usuario la interacción con la base de datos mediante el uso de ActiveRecord. Algunas de sus características fundamentales son las siguientes:

- ✓ Es Compatible con PHP4 y PHP5.
- ✓ Permite el uso de las funciones básicas CRUD (Create, Retrieve, Update y Delete en inglés).
- ✓ URLs amigables.
- ✓ Sistema de plantillas rápido y flexible.
- ✓ Helpers para AJAX, Javascript, HTML, forms, entre otros.
- ✓ Validación integrada.
- ✓ Scaffolding de las aplicaciones.
- ✓ Componentes de seguridad y sesión.

Struts

Es una herramienta de soporte para el desarrollo de aplicaciones Web bajo la plataforma J2EE (Java 2, Enterprise Edition). Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

Struts permite reducir el tiempo de desarrollo. Su carácter de "*software libre*" y su compatibilidad con todas las plataformas en que Java Entreprise esté disponible, lo convierte en una herramienta altamente disponible.

Cuando se programan aplicaciones Web usando el patrón MVC, en ocasiones surge la duda de usar un solo controlador o usar varios controladores, pues si se considera mejor usar un solo controlador para tener toda la lógica de control en un mismo lugar, se presenta un grave problema, ya que el controlador se convierte en lo que se conoce como "fat controller", es decir un controlador de peticiones. Struts surge como la solución a este problema ya que implementa un solo controlador que evalúa las peticiones del usuario mediante un archivo configurable (struts-config.xml).

Kumbia

Kumbia es un framework de uso libre bajo la licencia GNU/GPL. Constituye un esfuerzo por producir un framework que ayude a reducir el tiempo de desarrollo de una aplicación web sin producir efectos sobre los programadores.

Dentro de sus principales características se destacan:

- ✓ Sistema de Plantillas sencillo

- ✓ Administración de Cache
- ✓ Scaffolding Avanzado
- ✓ Modelo de Objetos y Separación MVC
- ✓ Soporte para AJAX
- ✓ Generación de Formularios
- ✓ Componentes Gráficos
- ✓ Seguridad

Está basado en los siguientes conceptos:

- ✓ Fácil de aprender.
- ✓ Fácil de instalar y configurar.
- ✓ Compatible con muchas plataformas.
- ✓ Listo para aplicaciones comerciales.
- ✓ Simple en la mayor parte de casos pero flexible para adaptarse a casos más complejos.
- ✓ Soportar muchas características de Aplicaciones Web actuales.
- ✓ Soportar las prácticas y patrones de programación más productivos y eficientes.
- ✓ Producir aplicaciones fáciles de mantener.
- ✓ Basado en Software Libre.

El principal principio es producir aplicaciones que sean prácticas para el usuario final y no sólo para el programador. La mayor parte de tareas que le quiten tiempo al desarrollador deberían ser automatizadas por Kumbia para que él pueda enfocarse en la lógica de negocio de su aplicación.

Es compatible con los tres sistemas Gestores de base de Datos disponibles más usados en la actualidad:

- ✓ MySQL.
- ✓ PostgreSQL.
- ✓ Oracle.

El modelo de objetos de Kumbia es utilizado en tres capas diferentes:

- ✓ Abstracción de la base de datos.
- ✓ Mapeo Objeto-Relacional.

- ✓ Modelo MVC (Modelo, Vista, Controlador).

Tiene en cuenta el uso otros proyectos de software libre para complementar su funcionalidad, entre ellos:

- ✓ FPDF: Reportes en formato PDF
- ✓ Prototype: Javascript crossbrowser
- ✓ Scriptaculous: Efectos visuales
- ✓ PHPMailer: Correo Electrónico
- ✓ Smarty: Motor de Plantillas potente y fácil de usar

1.2.8 Patrones de diseño

Los patrones de diseño permiten dar flexibilidad y extensibilidad a los diseños, ya que ellos no sólo nombran, abstraen e identifican aspectos claves de estructuras comunes de diseño, sino que generalmente son descritos en una forma específica documental, facilitando su comprensión y aplicación por los desarrolladores. Además contribuyen a reutilizar diseño, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones.

Patrones creacionales

Abstract Factory (Fábrica abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.

Builder (Constructor virtual): Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.

Factory Method (Método de fabricación): Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear.

Prototype (Prototipo): Crea nuevos objetos clonándolos de una instancia ya existente.

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Patrones Estructurales

Adapter (Adaptador): Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.

Bridge (Puente): Desacopla una abstracción de su implementación.

Composite (Objeto compuesto): Permite tratar objetos compuestos como si de uno simple se tratase.

Decorator (Envoltorio): Añade funcionalidad a una clase dinámicamente.

Facade (Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

Flyweight (Peso ligero): Reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.

Proxy : Mantiene un representante de un objeto.

Patrones de Comportamiento

Chain of Responsibility (Cadena de responsabilidad): Permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada.

Command (Orden): Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.

Interpreter (Intérprete): Dado un lenguaje, define una gramática para dicho lenguaje, así como las herramientas necesarias para interpretarlo.

Iterator (Iterador): Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.

Mediator (Mediador): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.

Memento (Recuerdo): Permite volver a estados anteriores del sistema.

Observer (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.

State (Estado): Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno.

Strategy (Estrategia): Permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución.

Template Method (Método plantilla): Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos, esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.

Visitor (Visitante): Permite definir nuevas operaciones sobre una jerarquía de clases sin modificar las clases sobre las que opera.

Patrones generales de software para asignar responsabilidades (GRASP)

Estos son algunos de los patrones GRASP más usados.

Patrón	Descripción
Experto	¿Quién asumirá la responsabilidad en el caso general? Asignar una responsabilidad al experto en información: la clase que posee la información necesaria para cumplir con la responsabilidad.

Creador	¿Quién crea? Asignar a la clase B la responsabilidad de crear una instancia de la clase A, si se cumple una de las siguientes condiciones: 1. B contiene. A 2. B agrega A 3. B tiene los datos de inicialización de A 4. B registra A 5. B utiliza A muy de cerca
Bajo Acoplamiento	¿Cómo dar soporte a poca dependencia y a una mayor reutilización? Asignar las responsabilidades de modo que se mantenga el bajo acoplamiento.
Alta Cohesión	¿Cómo mantener controlable la complejidad? Asignar las responsabilidades de modo que se mantenga una alta cohesión.

1.2.9 Ajax

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma. (25)

1.3 Herramientas, tecnologías, plataforma y librerías utilizadas en desarrollo del módulo

El uso de estas herramientas responde en primer lugar a la política llevada a cabo por la Facultad 7 de la Universidad de la Ciencias Informáticas y en particular por el Área Temática Sistemas Especializados perteneciente a dicha facultad. Esa política establece que las aplicaciones desarrolladas para el Ministerio de Salud Pública sean implementadas haciendo uso de herramientas y software libre. Para la selección de esas herramientas fue creado en esa facultad un Grupo de Arquitectura, el cual se encarga crear y divulgar el documento de arquitectura que contiene las herramientas y la arquitectura a utilizar por todos los proyectos productivos de la facultad. A continuación se justifica el uso de estas herramientas para el desarrollo de la aplicación.

1.3.1 Lenguajes de programación Web

Existen muchos lenguajes para el desarrollo de aplicaciones Web. Estos lenguajes se dividen en lenguajes de lado servidor (son aquellos lenguajes que son reconocidos, ejecutados e interpretados

por el propio servidor y que se envían al cliente en un formato comprensible para él) y lenguajes de lado cliente (son aquellos que pueden ser directamente interpretados por el navegador y no necesitan un pre-tratamiento, como son el HTML, el Java y el Java Script, los cuales son simplemente incluidos en el código HTML)
A continuación se describen los lenguajes utilizados para el desarrollo de la presente aplicación web.

PHP

Se utilizó PHP5 ya que la política del país aboga por la utilización del software libre, es multiplataforma y el mismo cumple con ese importante requisito, además es un lenguaje de programación interpretado usado normalmente para la creación de páginas web dinámicas.

El principal objetivo de PHP5 ha sido mejorar los mecanismos de POO para solucionar las carencias de las anteriores versiones. Un paso necesario para conseguir que PHP sea un lenguaje apto para todo tipo de aplicaciones y entornos, incluso los más exigentes. Entre sus ventajas principales se encuentra:

- ✓ Es un lenguaje multiplataforma.
- ✓ Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL
- ✓ Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- ✓ Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- ✓ Mejor soporte para la Programación Orientada a Objetos (que en versiones anteriores era extremadamente rudimentario) con PHP Data Objects.
- ✓ Biblioteca nativa de funciones sumamente amplia e incluida
- ✓ No requiere definición de tipos de variables.
- ✓ Tiene manejo de excepciones.
- ✓ Mejoras de rendimiento.

- ✓ Mejor soporte para MySQL con extensión completamente reescrita.
- ✓ Mejor soporte a XML (XPath, DOM, etc.).
- ✓ Soporte nativo para SQLite.
- ✓ Soporte integrado para SOAP.

Java Script

Se utiliza JavaScript por sus ventajas y funcionalidades como son: no requiere un tiempo de compilación, los scripts se pueden desarrollar en un periodo de tiempo relativamente corto. A esto se puede añadir las características de interfaz como, por ejemplo, cuadro de diálogo, formularios y otros elementos GUI (Interfaz Gráfico de Usuario), son gestionados por el navegador y por el código HTML. Por lo tanto los programadores que utilizan JavaScript no se deben preocupar en crear o controlar dichos elementos en sus aplicaciones.

Aunque JavaScript tiene muchas similitudes con Java, no incluye la sintaxis y reglas complejas de Java. Como WWW es independiente de la plataforma hardware o sistema operativo, los programas escritos en JavaScript también lo son, siempre y cuando exista un navegador con soporte JavaScript para la plataforma en cuestión.

Los programas JavaScript tienden a ser pequeños y compactos, no requieren mucha memoria ni tiempo adicional de transmisión. Además, al incluirse dentro de las mismas páginas HTML se reduce el número de accesos independientes a la red.

Permite mejorar el aspecto y la funcionalidad de las páginas web, además brinda la posibilidad de realizar validaciones en la entrada de datos.

1.3.2 Entorno de desarrollo integrado (IDE)

Zend Studio

Se decidió utilizar Zend Studio como IDE de desarrollo ya que se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web en lenguaje PHP. Es un programa multiplataforma, que además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código.

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. (26)

1.3.3 Gestor de Base de Datos

MySQL

Se utiliza MySQL debido a que tiene la ventaja de ser un Sistema Gestor de Base de Datos robusto, que puede almacenar gran cantidad de datos, es rápido, seguro, estable, multiplataforma, gratuito y soporta múltiples lenguajes de programación. Otras de sus ventajas son:

- ✓ Escalabilidad: es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla.
- ✓ MySQL está escrito en C y C++ y probado con multitud de compiladores y dispone de APIs para muchas plataformas diferentes.
- ✓ Conectividad: es decir, permite conexiones entre diferentes máquinas con distintos sistemas operativos. Es corriente que servidores Linux o Unix, usando MySQL, sirvan datos para ordenadores con Windows, Linux, Solaris, etc. Para ello se usa TCP/IP, tuberías, o sockets Unix.
- ✓ Es multihilo, con lo que puede beneficiarse de sistemas multiprocesador.
- ✓ Permite manejar multitud de tipos para columnas.
- ✓ Permite manejar registros de longitud fija o variable.

1.3.4 Cascading Style Sheets (CSS)

Las Hojas de Estilo en Cascada se utilizan en la aplicación ya que es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser mostrada la información presente en ese documento a través de un dispositivo de lectura.

Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

Se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los Estilos definen la forma de mostrar los elementos HTML y XML. Permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a ella. (27)

1.3.5 Lenguaje de modelado

UML

Como lenguaje de modelado se utiliza UML ya que su función es describir y documentar un sistema y entre sus principales ventajas está:

- ✓ Mejores tiempos totales de desarrollo (de 50% o más).
- ✓ Con el uso de UML las fases de análisis y diseño consumirán mayor tiempo, pero el tiempo de construcción, implantación y estabilización se reducen drásticamente debido a que no hay correcciones mayores en las fases de mayor impacto de un proyecto.
- ✓ El mantenimiento correctivo se reduce drásticamente (hasta un 80% con respecto a un sistema hecho sin metodología).
- ✓ Mejor soporte a la planeación y al control de proyectos.
- ✓ Mayor independencia del personal de desarrollo. Al tener documentadas las aplicaciones en un lenguaje estándar.
- ✓ Mayor soporte al cambio organizacional, comercial y tecnológico. Un modelo permite cuantificar el impacto de un cambio antes de hacerlo y permite ensayar distintos enfoques de solución.
- ✓ Alto re-uso, los productos de un desarrollo pueden ser usados en otro. Se pueden crear componentes reusables que con la difusión y administración adecuadas minimizarán costos y errores.
- ✓ Minimización de costos. Los puntos antes mencionados tienen un impacto económico que generalmente tiende a ser proporcional al tamaño de la organización.

1.3.6 Framework de desarrollo

Framework CodeIgniter

Se tomo la decisión de usar CodeIgniter como plataforma de desarrollo en la aplicación ya que es un framework para desarrollo de aplicaciones en PHP. Es de código abierto, multiplataforma, tiene una interface simple y un acceso a sus librerías bien estructurado.

CodeIgniter es otro framework para PHP, una alternativa a otros que hay disponibles. Es adecuado para desarrollos que no requieran un framework que marque mucho la aplicación, también cuando sea necesario mucho rendimiento. Se destaca en:

- ✓ Bajo uso de recursos.
- ✓ Rendimiento excepcional.
- ✓ Altamente compatible con gran variedad de versiones y configuraciones de PHP.

Conclusiones

Con el único propósito de desarrollar una aplicación con la calidad requerida y que sea un eslabón más dentro de la informatización de la salud cubana, en este capítulo se analizaron y valoraron las principales tecnologías utilizadas para el desarrollo de aplicaciones web en el mundo, y las utilizadas para el desarrollo del módulo Salud Ambiental del Programa Control Sanitario Internacional del Ministerio de Salud Pública, que en su mayoría, cumplen con la política del país respecto a la utilización de software libre.

CAPÍTULO 2: ELEMENTOS DE ARQUITECTURA

En este capítulo se presentan los elementos de arquitectura utilizados para llevar a cabo el desarrollo del Módulo salud Ambiental. Se parte de una serie de elementos conceptuales relacionados con la Arquitectura de Software, luego se exponen los requerimientos no funcionales que debe cumplir sistema propuesto para su correcto funcionamiento. Además se explican los patrones o estilos arquitectónicos que sirvieron de guía para el desarrollo del software. Por último y no menos importante, se muestra el modelo de implementación compuesto por el diagrama de componentes y la vista de despliegue.

2.1 Selección y argumentación de los requerimientos no funcionales del sistema propuesto

Los requerimientos no funcionales responden a cualidades que el producto debe tener y las características para que este sea atractivo, confiable, usable y seguro. No se refieren a funciones específicas que proporciona el sistema. Son restricciones de los servicios o funciones ofrecidas por el sistema y surgen de las necesidades del usuario.

El hecho de que sea “no funcional” no significa que deje de tener funcionalidad, debido a que puede comprometer el funcionamiento del producto desarrollado, y tiene como premisa que, una vez conocido lo que el sistema debe hacer, se determine cómo ha de comportarse y qué cualidades debe tener el mismo.

Forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con la toda la funcionalidad requerida, las propiedades no funcionales pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

También pueden referirse al proceso de desarrollo: uso de una herramienta CASE, lenguaje de programación o metodología concreta.

A continuación se detallan los requerimientos no funcionales de la aplicación desarrollada:

2.1.1 Requerimientos de Usabilidad

- ✓ Debe lograr que las interacciones del usuario con el sistema sean predecibles y familiares.
- ✓ Debe posibilitar múltiples vías por las cuales el usuario pueda realizar una tarea.
- ✓ La aplicación debe ser flexible y permitir que el usuario aprenda a usarla con facilidad.
- ✓ La aplicación podrá ser usada por cualquier persona que posea conocimientos básicos en computación y en ambientes Web.
- ✓ Debe brindar la posibilidad de diálogos, con el objetivo de mantener todo el tiempo orientado al usuario.

2.1.2 Requerimientos Soporte

- ✓ El sistema estará bien documentado para garantizar futuros mantenimientos.
- ✓ Se le debe dar mantenimiento periódico a los servidores de bases de datos controlando la integridad de la información.

2.1.3 Requerimientos de Ayuda y Documentación

- ✓ Contará con un manual de usuarios para que se pueda explotar al máximo.
- ✓ Contará con una ayuda digital, a la cual se podrá acceder desde cualquier parte de la aplicación.

2.1.4 Requerimientos de Confiabilidad

- ✓ El sistema estará disponible las 24 horas del día, tanto para el trabajo de los usuarios como para las acciones de mantenimiento.
- ✓ Deberá prevenir los posibles fallos y/o errores que pudieran presentarse y posibilitar una rápida recuperación en dichos casos.

2.1.5 Requerimientos de Software

- ❖ **Para el cliente:**

- ✓ Tendrá acceso al sistema a través de cualquier navegador Web. Se recomienda: Internet Explorer 4.0 o superior y Mozilla 1.5 o superior.
- ✓ Sistema Operativo Linux o Windows 98 ó Superior.

❖ **Para el servidor:**

- ✓ Sistema operativo Linux.
- ✓ Servidor Web Apache 2.0.
- ✓ PHP 5.
- ✓ Framework CodeIgniter.
- ✓ Servidor de Base de Datos MySQL 5.0.

2.1.6 Requerimientos de Hardware

❖ **Para el cliente:**

- ✓ Procesador Pentium III o superior.
- ✓ 256 de memoria RAM o superior.
- ✓ Monitor VGA o superior.
- ✓ Tarjeta de red.

❖ **Para el servidor:**

- ✓ Procesador Pentium IV o superior.
- ✓ 2 Gb de memoria RAM.
- ✓ Disco Duro de 80 GB o superior.
- ✓ Monitor VGA o superior.
- ✓ Tarjeta de red.

2.1.7 Requerimientos de Interfaz

- ✓ Se podrán distinguir colores atractivos y acordes con los recomendados para los software de salud.
- ✓ Debe poseer un ambiente amigable, intuitivo, sencillo y de fácil navegación, tratando así de impedir el rechazo por parte del usuario al tener que interactuar con un sistema no conocido.
- ✓ Paginación de reportes de búsqueda, y listados.
- ✓ Diseño perfectamente encuadrado para resoluciones de 1024 x 768, pero preparado para verse en otras resoluciones.

2.1.8 Requerimientos de Seguridad

- ✓ La autenticación de los usuarios en el sistema, será garantizada por el Sistema de Autentificación, Autorización y Auditoría (SAAA) al cual estará conectada la aplicación.
- ✓ Se debe restringir las funcionalidades mediante roles de usuarios garantizando que la información sea accesible al usuario autorizado.
- ✓ La información deberá ser consultada en dependencia del nivel que ocupe el usuario en la escala administrativa del MINSAP.
- ✓ El sistema deberá estar protegido contra accesos no autorizados y las modificaciones de información.

2.1.9 Restricciones de Diseño e Implementación

- ✓ Utilizar los patrones de diseño GRASP.
- ✓ Para el análisis y el diseño del sistema debe ser utilizada la metodología RUP, usando el lenguaje de modelación UML y como herramienta para llevarlo a cabo el Visual Paradigm.
- ✓ Implementado con el lenguaje de programación PHP 5
- ✓ Desarrollado en Zend Studio.

2.1.10 Requerimientos de Portabilidad

- ✓ El producto podrá ser usado bajo cualquier sistema operativo ya sea Unix, Linux o Windows.

2.2 Patrones y estilos arquitectónicos

A continuación, se exponen dos conceptos sobre Arquitectura de Software planteados por personalidades que son considerados autoridades en el mundo de la Ingeniera de Software:

"… es una descripción de los subsistemas y los componentes de un sistema informático y las relaciones entre ellos (...)"

Roger Pressman

"… La arquitectura de un sistema constituye un amplio marco que describe su forma y su estructura, sus componentes y cómo estos interactúan (...)"

Jerrold Grochow

Los estilos de arquitectura sirven de guía para la organización del sistema de software. Estos incluyen reglas y líneas a seguir para la organización de un sistema. Un patrón es "una solución a un problema de diseño que aparece con frecuencia".

Los arquitectos moldean el sistema para darle una forma. Es esta forma, la arquitectura, la que debe diseñarse para permitir que el sistema evolucione, no sólo en su desarrollo inicial, sino también a lo largo de las futuras iteraciones. Para encontrar esa forma, los arquitectos deben trabajar sobre la comprensión general de las funciones claves, es decir, sobre los casos de uso claves del sistema.

Teniendo en cuenta que actualmente el país a puesto todo su empeño en impulsar la Industria del Software y que para lograr este propósito se requiere que los sistemas desarrollados se realicen con el menor costo y en el menor tiempo posible, siempre y cuando cumplan con los parámetros de calidad requeridos; se necesitan desarrollar componentes completamente reutilizables que permitan disminuir esos costos y tiempos de desarrollos. Para ello, el diseño de la arquitectura dentro del desarrollo de un software, juega un papel de suma importancia.

A continuación se expone una serie de elementos de arquitectura que se tuvieron en cuenta para el desarrollo de este sistema:

2.2.1 Modelo Vista Controlador

Se utiliza el patrón arquitectónico **Modelo Vista Controlador** o **MVC** ya que describe una forma, muy utilizada en la Web, de organizar el código de una aplicación separando los datos, la interfaz de usuario, y la lógica de control en tres componentes distintos. (28)

- ✓ El modelo, que contiene la lógica de negocio de la aplicación.
- ✓ La vista, que muestra al usuario la información que éste necesita.
- ✓ El controlador, que recibe e interpreta la interacción del usuario, actuando sobre el modelo y la vista de manera adecuada para provocar cambios de estado en la representación interna de los datos, así como en su visualización.

Si se realiza un diseño que mezcle los componentes de interfaz y de negocio, entonces el resultado será que, cuando se necesite cambiar el interfaz, se tendrá que modificar trabajosamente los componentes de negocio. Mayor trabajo y más riesgo de error.

Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. (29)

- ✓ Hay una clara separación entre los componentes de un programa; lo cual permite implementarlos por separado.
- ✓ Podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- ✓ La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

2.2.2 Desarrollo basado en componentes

El paradigma de ensamblar componentes y escribir código para hacer que estos componentes funcionen se conoce como Desarrollo de Software Basado en Componentes. El uso de este paradigma posee algunas ventajas: (30)

- ✓ **Reutilización del software**. Posibilita alcanzar un mayor nivel de reutilización de software.
- ✓ **Simplifica las pruebas**. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- ✓ **Simplifica el mantenimiento del sistema**. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- ✓ **Mayor calidad**. Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

De la misma manera, el optar por usar componentes de terceros en lugar de desarrollarlos, posee algunas ventajas: (31)

- ✓ **Ciclos de desarrollo más cortos**. La adición de una pieza dada de funcionalidad tomará días en lugar de meses ó años.
- ✓ **Mejor ROI**. Usando correctamente esta estrategia, el retorno sobre la inversión puede ser más favorable que desarrollando los componentes uno mismo.

- ✓ **Funcionalidad mejorada**. Para usar un componente que contenga una pieza de funcionalidad, solo se necesita entender su naturaleza, más no sus detalles internos. Así, una funcionalidad que sería impráctica de implementar, se vuelve ahora completamente asequible.

2.2.3 Arquitectura Cliente Servidor

El modelo Cliente/Servidor es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual a los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", se les da respuesta por otros computadores llamados servidores". (32)

Esta arquitectura es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes. (33)

Entre las principales características de esta arquitectura se pueden destacar las siguientes:

- ✓ El servidor presenta a todos sus clientes una interfaz única y bien definida.
- ✓ El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- ✓ El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- ✓ Los cambios en el servidor implican pocos o ningún cambio en el cliente.

❖ **Cliente**

En esta arquitectura el remitente de una solicitud es conocido como cliente. Sus características son:

- ✓ Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación.
- ✓ Espera y recibe las respuestas del servidor.
- ✓ Por lo general, puede conectase a varios servidores a la vez.
- ✓ Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Los clientes realizan generalmente funciones como:

- ✓ Manejo de la interfaz de usuario.
- ✓ Captura y validación de los datos de entrada.
- ✓ Generación de consultas e informes sobre las bases de datos.

❖ **Servidor**

Al iniciarse esperan a que lleguen las solicitudes de los clientes, por lo tanto desempeñan un papel pasivo en la comunicación.

- ✓ Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- ✓ Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- ✓ No es frecuente que interactúen directamente con los usuarios finales.

Por su parte los servidores realizan, entre otras, las siguientes funciones:

- ✓ Gestión de periféricos compartidos.
- ✓ Control de accesos concurrentes a bases de datos compartidas.
- ✓ Enlaces de comunicaciones con otras redes de área local o extensa.

❖ **Ventajas**

- ✓ Uno de los aspectos que más ha promovido el uso de sistemas Cliente/Servidor, es la existencia de plataformas de hardware cada vez más baratas. Además, se pueden utilizar componentes, tanto de hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones. (34)
- ✓ Centralización del control: Los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P).

- ✓ Escalabilidad: Se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
- ✓ Fácil mantenimiento: Al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.
- ✓ Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad del interfaz, y la facilidad de empleo.

En el caso del subsistema Salud Ambiental se usa esta arquitectura ya que la aplicación estará ubicada en un servidor central en Infomed, el cual se encargará de dar respuesta a las peticiones realizadas por los clientes que utilicen la aplicación.

2.2.4 Arquitectura en 3 capas

La **programación por capas** es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

La arquitectura de una aplicación es la vista conceptual de la estructura de esta, ya que la aplicación contiene código de presentación (Capa de Presentación), código de procesamiento de datos (Capa de negocio) y código de almacenamiento de datos (Capa de Datos). La arquitectura de las aplicaciones difiere según como esté distribuido este código.

El desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca el nivel requerido sin tener que revisar entre código mezclado. Permite distribuir el trabajo de creación de una aplicación por niveles. (35)

En el diseño actual de sistemas informáticos se suele usar las arquitecturas multinivel o **Programación por capas**. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten). (36)

- **Capas o niveles**

Capa de presentación: es la que ve el usuario (hay quien la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. (37)

Capa de negocio: residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos del mismo. (38)

Capa de datos: en esta residen los datos. Está formada por uno o más gestores de bases de datos que realiza(n) todo el almacenamiento de datos, recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador (no sería lo normal), si bien lo más usual es que haya una multitud de ordenadores donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor).

Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio. Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares. El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico: (39)

Presentación/ Lógica de Negocio/ Datos.

En cambio, el término "nivel", corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

Una solución de tres capas (presentación, lógica, datos) que residen en un solo ordenador (Presentación + lógica + datos). Se dice, que la arquitectura de la solución es de tres capas y un nivel.

Una solución de tres capas (presentación, lógica, datos) que residen en dos ordenadores (presentación + lógica, lógica + datos). Se dice que la arquitectura de la solución es de tres capas y dos niveles. Una solución de tres capas (presentación, lógica, datos) que residen en tres ordenadores (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y tres niveles.

Tiene las siguientes ventajas:

- ✓ Desarrollos paralelos. El desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de tal forma que basta con conocer la forma de comunicación existente entre los niveles.
- ✓ Aplicaciones más robustas debido al encapsulamiento.
- ✓ Mantenimiento y soporte más sencillo. Es más sencillo cambiar un componente que modificar una aplicación monolítica).
- ✓ Mayor flexibilidad. Se pueden añadir nuevos módulos para dotar al sistema de una nueva funcionalidad.
- ✓ Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

2.3 Modelo de Implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios.

Este artefacto describe cómo se implementan los componentes, congregándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre éstos.

2.3.1 Vista de Despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, este no es más que la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software. Estos estarán formados por instancias de los componentes software que representan manifestaciones del código en tiempo de ejecución.

Es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos y procesos.

Los diagramas de despliegue son fundamentalmente diagramas de clases que se ocupan de modelar los nodos de un sistema. Los nodos se conectan mediante asociaciones de comunicación. Estas asociaciones indican:

- ✓ Algún tipo de ruta de comunicación entre los nodos.
- ✓ Los nodos intercambian objetos o envían mensajes a través de esta ruta.
- ✓ El tipo de comunicación se identifica con un estereotipo que indica el protocolo comunicación o la red.

El módulo Salud Ambiental (SA) estará desplegado de la siguiente manera:

El código fuente de la aplicación se alojará en un servidor web ubicado en Infomed al cual el usuario accederá a través de un navegador web mediante el protocolo http, haciéndole peticiones y recibiendo respuesta del mismo desde una computadora cliente, la cual a su vez se encontrará conectada a una impresora para imprimir reportes y datos útiles para el usuario.

El servidor del SiSalud se comunicará con el servidor de aplicación SA haciendo uso de servicios web XML y mediante el protocolo SOAP para brindar los servicios definidos que intervienen en el correcto funcionamiento de la aplicación.

Esta contará también con un servidor de base de datos que establecerá una comunicación mediante el protocolo TCP/IP para el obtener los datos y ser mostrados al cliente.

La siguiente figura muestra el diagrama de despliegue del Módulo Salud Ambiental del Proyecto Control Sanitario Internacional:

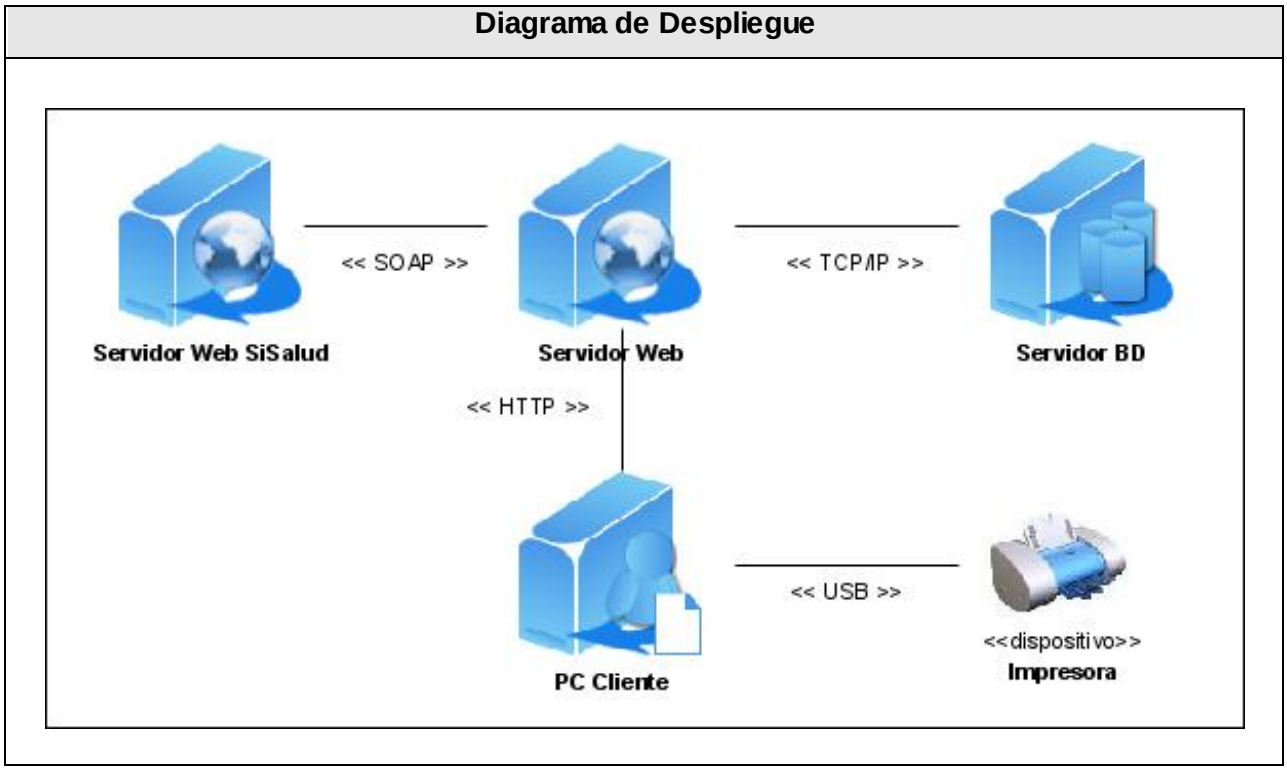


Figura 2.1 Diagrama de Despliegue.

2.3.2 Vista de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes *software*, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del *software*, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. En cuanto a los componentes, sólo aparecen tipos de componentes, ya que las instancias específicas de cada tipo se encuentran en el diagrama de despliegue. Dado que los diagramas de componentes muestran los componentes *software* que constituyen una parte reusable, sus interfaces, y sus

interrelaciones, en muchos aspectos se puede considerar que un diagrama de componentes es un diagrama de clases a gran escala.

Cada componente en el diagrama debe ser documentado con un diagrama de componentes más detallado, un diagrama de clases, o un diagrama de casos de uso. (40)

En la siguiente figura se muestra de forma global los diferentes subsistemas de implementación en los que se divide la aplicación. Se puede observar como el subsistema Salud Ambiental se divide en en tres subsistemas: Vistas, Controladoras y Modelos. Por otro lado el subistema Modelos se encarga de interactuar con los componentes existentes en el subsistema SISalud y con la base de datos del módulo.

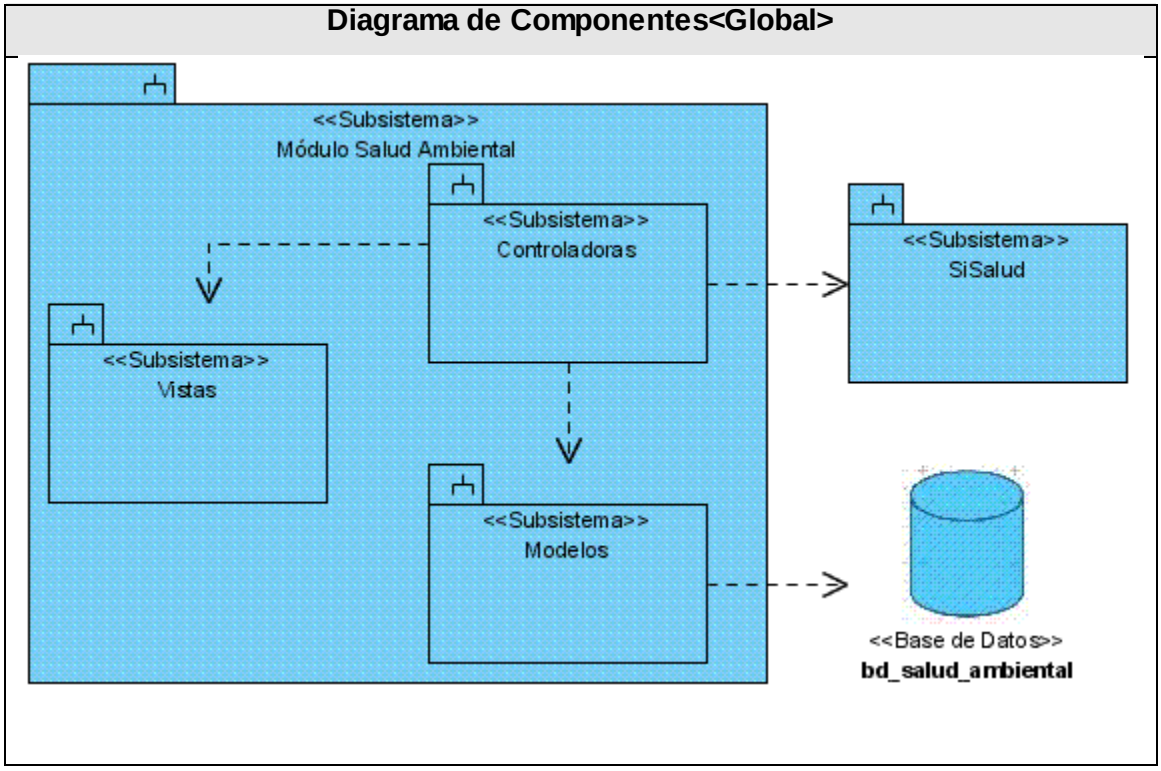


Figura 2.2 Diagrama Componentes (Global)

A continuación se muestra el subsistema de las **Vistas**, el cual se encuentra organizado en cuatro subsistemas.

Vistas Registro Sanitario: contiene las vistas relacionadas con las funcionalidades del registro sanitario de un producto.

Vistas Configuración Registro Sanitario: como su nombre lo indica, contiene las vistas relacionadas con las funcionalidades que permiten configurar los nomencladores del registro sanitario de un producto.

Vistas Salud Ambiental: este subsistema esta compuesto por tres subsistemas.

- ✓ **Vistas Cargas:** agrupa las vistas relacionadas con la gestión de los datos de las cargas de un producto determinado.
- ✓ **Vistas Extracciones:** incluye las vistas relacionadas con la Autorización de Extracción de un producto.
- ✓ **Vistas Inspecciones:** contiene las vistas que intervienen en las inspecciones sanitarias realizadas a los productos.

Vistas Configuración Salud Ambiental: esta formado por las vistas relacionadas con la configuración de los nomencladores de Salud Ambiental.

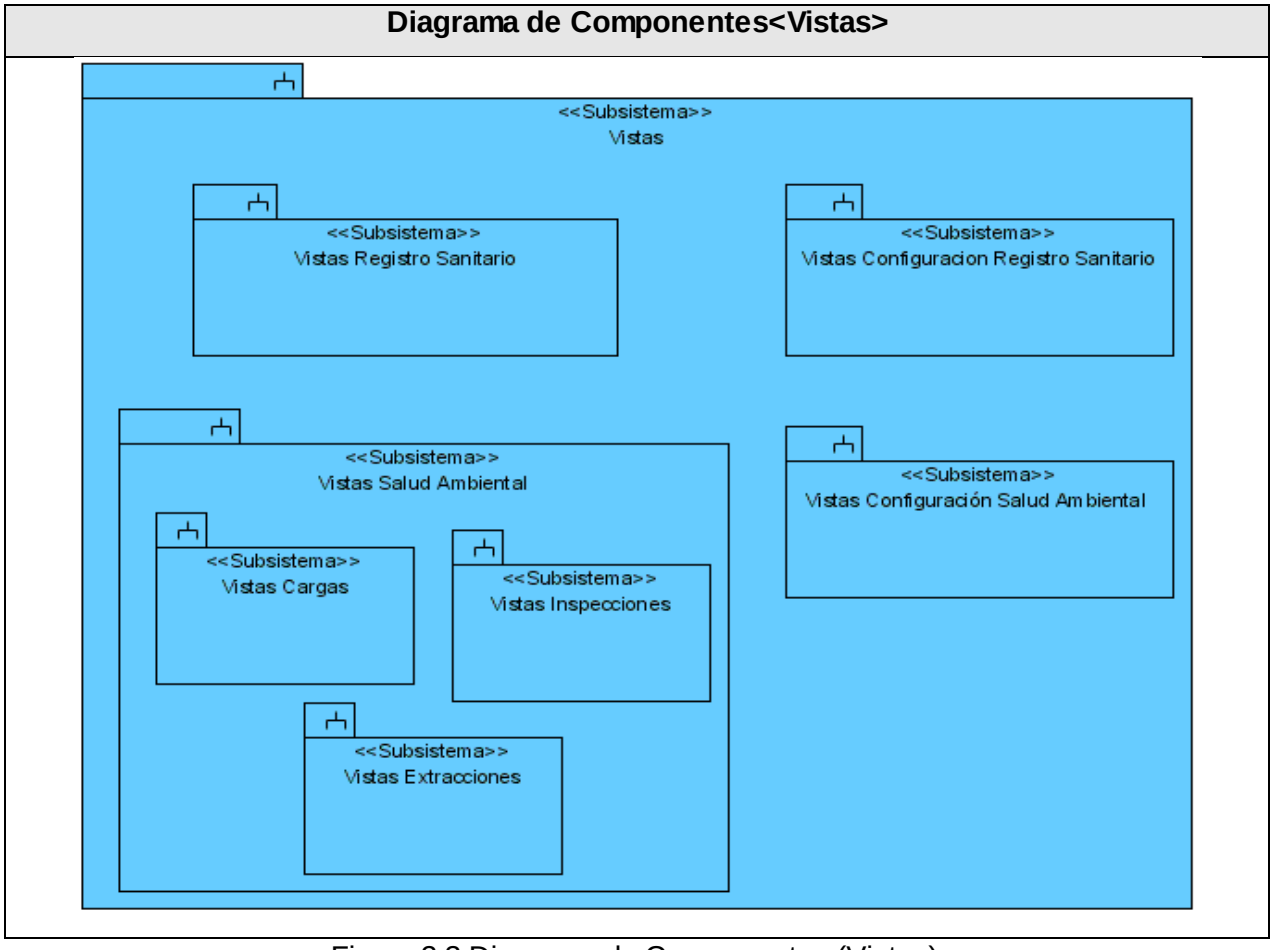


Figura 2.3 Diagrama de Componentes (Vistas).

Esta figura muestra la relación existente entre los componentes del subsistema Controladoras y los componentes del subsistema Modelos.

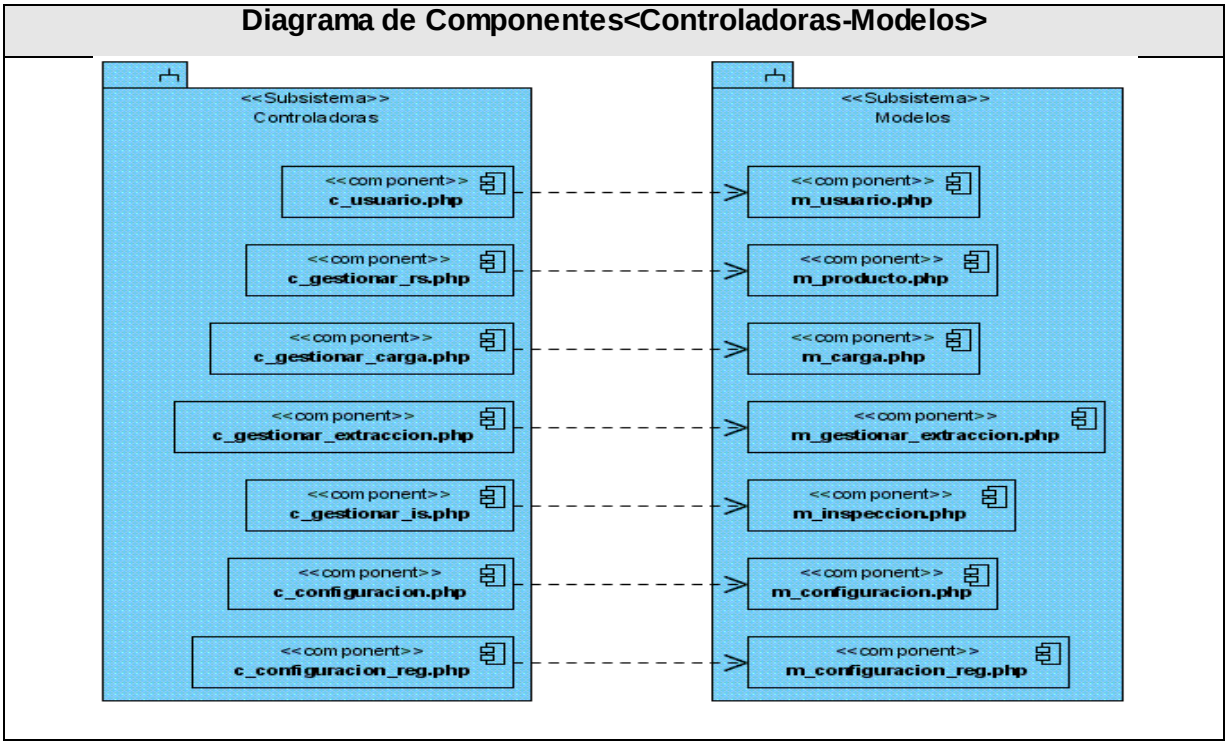


Figura 2.4 Diagrama de Componentes (Controladoras - Modelos).

El diagrama que se muestra a continuación contiene la relación entre los componentes que integran el subsistema Vistas Cargas y el componente que se encarga del control de esas vistas.

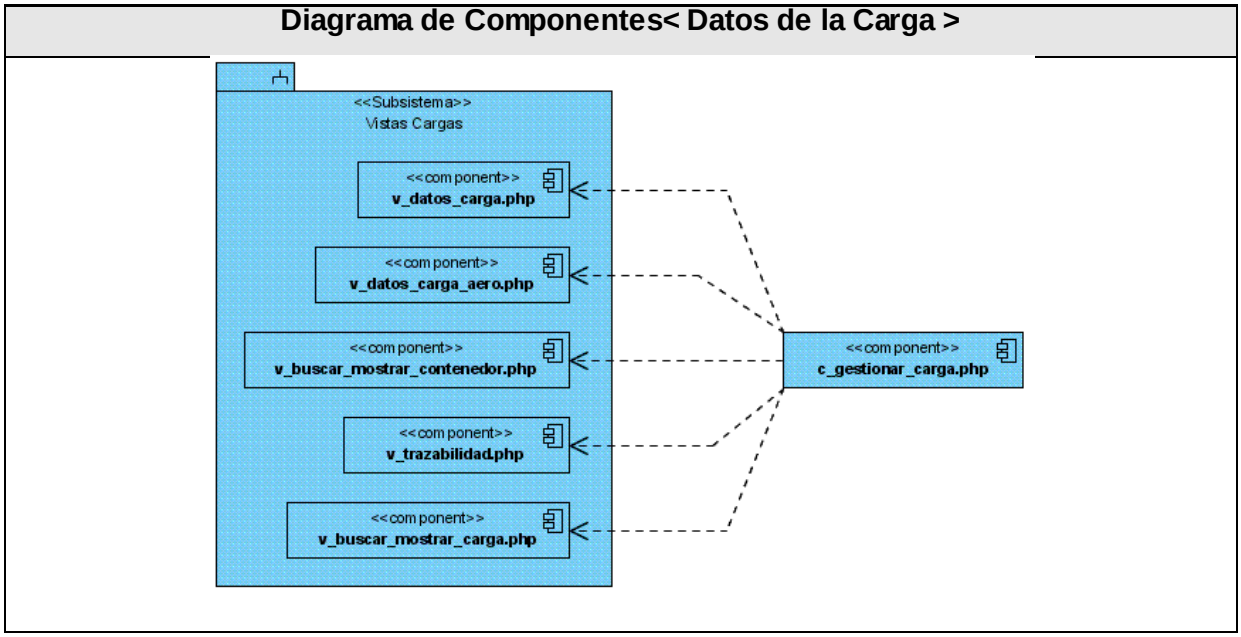


Figura 2.5 Diagrama de Componentes (Datos de la Carga).

En este diagrama se muestra la relación entre los componentes que integran el subsistema Vistas Extracciones y el componente encargado del control de esas vistas.

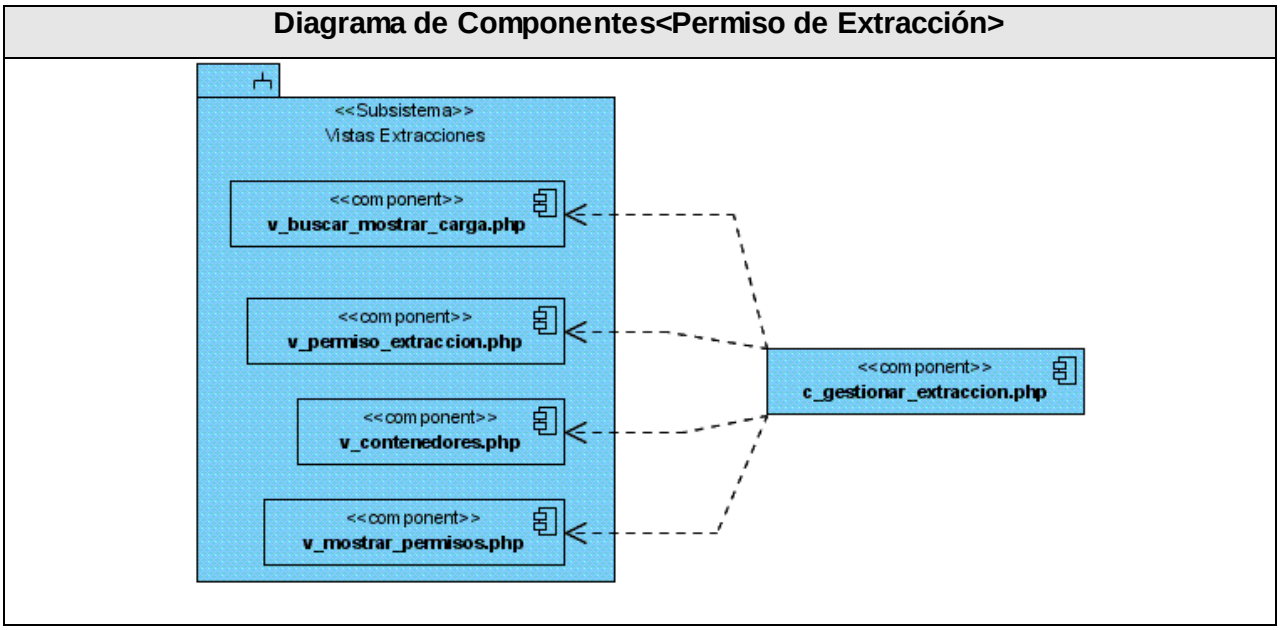


Figura 2.6 Diagrama de Componentes (Permiso de Extracción).

Este último muestra la relación existente entre los componentes que integran el subsistema Vistas Inspección y el componente que se encarga del control de estas vistas.

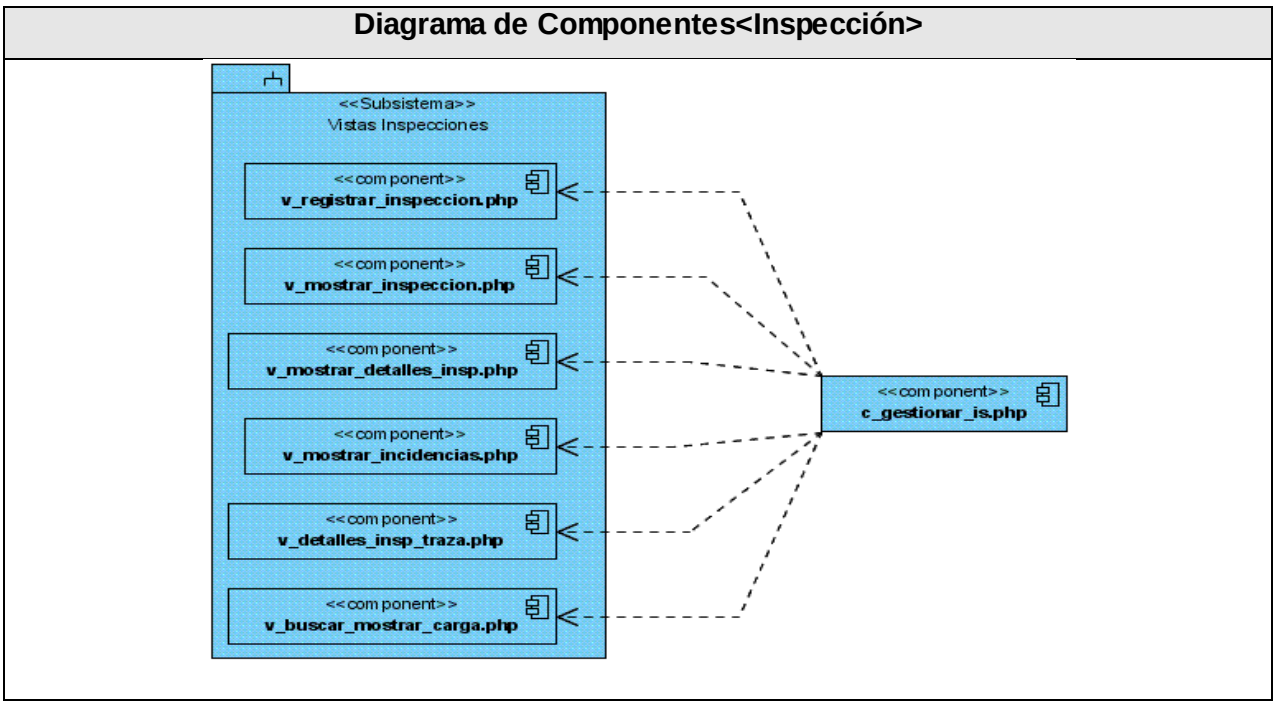


Figura 2.7 Diagrama de Componentes (Inspeccion).

Conclusiones

En este capítulo se presentaron los elementos de arquitectura utilizados para el desarrollo de la aplicación, donde se destacan los requisitos no funcionales y los patrones arquitectónicos presentes en el sistema propuesto. Además se elaboró el diagrama de implementación y la vista de despliegue, que describen como estará dividida la aplicación en componentes y la relación física entre estos una vez desplegada la aplicación.

CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

En este capítulo se lleva a cabo la descripción y el análisis de la solución propuesta para el módulo de Salud Ambiental. Se toma como punto de partida el análisis de los componentes o módulos ya existentes en el Sistema de Información para la Salud que fueron definidos para ser usados. Luego se realiza una descripción detallada de las tablas de la base de datos, así como de las clases utilizadas en la implementación de la aplicación.

3.1 Estrategias de integración con otros componentes o módulos

En el mundo actual los rápidos cambios surgidos en los sistemas de información de las organizaciones han disminuido el tiempo y los recursos asignados para la realización de proyectos de software. Para adaptarse a estos nuevos retos, la Ingeniería del Software ha evolucionado buscando la generación de sistemas software de mayor calidad, con menores costes y tiempos de desarrollo, en los cuales la rapidez en el desarrollo no comprometa la calidad de las soluciones creadas.

En este contexto, la reutilización de componentes de software previamente desarrollados y probados por otros se ha definido como una línea de trabajo prioritaria en el desarrollo de aplicaciones. La posibilidad de usar los servicios web brindados por otros componentes garantiza la reutilización de los datos y evita la duplicidad de la información. Aunque en ocasiones la falta de documentación sobre el componente que se desea usar atenta contra el uso óptimo del mismo y la calidad del sistema resultante.

Como se definió anteriormente para el desarrollo del presente sistema se hizo uso de la arquitectura basada en componentes, donde cada componente es un módulo con funcionalidades propias que utiliza los servicios brindados por otros componentes.
Por tal motivo el funcionamiento del módulo Salud Ambiental depende de otros componentes existentes en Sistema de Información para las Salud, los cuales se describen a continuación.

3.1.1 Componente de Seguridad (SAAA)

Este componente tiene implementado un mecanismo de seguridad basado en el modelo de Autenticación, Autorización y Auditoria. La Autenticación es la primera acción del usuario en el sistema y consiste en suministrar un nombre de usuario único y una contraseña. Si el usuario no se encuentra registrado se reportará un error de acceso.

Si el usuario autenticado se encuentra registrado se autorizará su acceso y se crea un certificado digital que contiene un identificador único (token) de 32 caracteres, que tiene como información el identificador del usuario, el nivel de acceso (Nacional, Provincial, Municipal o Unidad de Salud), el identificador de nivel de acceso, un listado de los módulos a los que el usuario tiene acceso y el tipo de acceso de cada uno de ellos (Editor o Visualizador).

El certificado de seguridad es utilizado para acceder al resto de los componentes brindados por el SISalud. La autenticación en el módulo Salud Ambiental se implementó utilizando el SAAA debido a la política de integración que deben cumplir todos los sistemas informáticos producidos en la UCI para el Ministerio de Salud Pública.

De este componente solo se utiliza la autenticación ya que la autorización es manejada por la aplicación a través de roles, donde un usuario pertenece a un rol y un rol agrupa un conjunto de usuarios con privilegios comunes en el sistema.

En el SAAA un usuario tiene una lista de módulos a los que puede acceder, por tal motivo no satisface necesidades en cuanto a la autorización dentro del módulo Salud Ambiental (SA). Los privilegios manejados en el módulo SA son a nivel de método o función, es decir, un rol tiene acceso a determinadas funciones contenidas en una clase específica de la aplicación.

La información que se necesita de este servicio se muestra a continuación:

Componente	***Servicios Consumidos***	***Datos Utilizados***	***Justificación***
Sistema de Seguridad (SAAA)	Autenticar	• Idusaurio • Idusuarioexterno • Nombre	Con el objetivo de conocer el usuario que está autenticado

		• Certificado
		• Nivel
		• Idnivel
		• ListaDerecho
		  ○ Nombremodulo
		  ○ Modulo
		  ○ Tipousuario
		  ○ url
			en el sistema, a qué nivel pertenece, qué tipo de usuario es y a qué módulos tiene acceso.

3.1.2 Registro de Ubicación (RU)

El Registro de Ubicación es uno de los componentes no médicos del Registro Informatizado de Salud. El mismo se encarga de gestionar la información relacionada con las provincias, municipios, localidades, calles y manzana existentes en el país. Los servicios brindados por este componente son utilizados por el módulo Salud Ambiental para registrar la ubicación de los centros que constituyen el destino de las cargas por contenedores que entran por los puertos y aeropuertos del país.

En la siguiente tabla se muestran los servicios consumidos de este componente y su justificación.

Componente	***Servicios Consumidos***	***Datos Utilizados***	***Justificación***

Registro de Ubicación (RU).	ListarProvincias	• Provincias
		  ○ Id
		  ○ descri (descripción)
			Para listar todas las provincias.

	ListarMunicipio	• Idmunicipio
		• Nombre
		• Provincia
		  ○ Idprovincia
		  ○ nombre
		• totalLocalidades
		• localidades
			Para listar los municipios y para listar las localidades de cada municipio.

		○ idLocalidad ○ Localidad	
	BuscarLocalidad	• ArrayStructLocalidad ○ IdIocalidad ○ Localidad ○ Idmunicipio ○ Idprovincia ○ Municipio ○ provincia	Para listar las localidades por municipio.
	ListarCallesPor-Localidad	• structcalle1 ○ id_calles ○ descripción ○ idmunicipio ○ idprovincia ○ idlocalidad ○ municipio ○ provincia ○ localidad	Para listar las calles por localidad, en caso no necesite hacerlo por manzana.

3.2 Descripción de las clases u operaciones necesarias

Para darle cumplimiento a las funcionalidades que requiere el sistema se implementaron 7 clases controladoras y 8 clases modelos. A continuación se describen aquellas que se consideran críticas debido a que representan procesos fundamentales en lo que a negocio se refiere. Las otras que no se consideran críticas se describen en el ***Anexo 2*. (Descripción de clases de configuración y seguridad)**.

Nombre: C_Gestionar_RS	
Tipo de clase: controladora	
Atributo:	**Tipo**
Para cada responsabilidad:	
Nombre:	Mostrar_Otorgar_Registro()
Descripción:	Permite mostrar la vista usada para otorgar un registro sanitario.
Nombre:	Cargar_Nomencladores()
Descripción:	Permite obtener los listados de nomencladores.
Nombre:	Adicionar()
Descripción:	Permite adicionar un producto con sus especificaciones.
Nombre:	Mostrar_Buscador()
Descripción:	Permite mostrar el buscador de productos.
Nombre:	Cargar($id, $accion, $vista)
Descripción:	Permite buscar los datos de un producto dado su id y muestra el resultado en la vista que recibe como parámetro.
Nombre:	Cargar2()
Descripción:	Permite buscar los datos de un producto dado su id, muestra el resultado en una vista con los campos editables o no editables.
Nombre:	Buscar_Producto()
Descripción:	Permite buscar los datos de un producto usando 5 criterios de búsqueda.
Nombre:	Modificar()

Descripción:	Permite modificar los datos de un producto y sus especificaciones.
Nombre:	Eliminar()
Descripción:	Permite eliminar un producto y sus especificaciones.
Nombre:	Listar_Pendientes()
Descripción:	Se encarga de listar los productos que están pendientes de registro.
Nombre:	Registrar_Pendientes()
Descripción:	Se encarga de registrar los productos que están pendientes de registro.
Nombre:	Ir_a_Modificar()
Descripción:	Se encarga de mandar la acción (Modificar) hacia el buscador.
Nombre:	Ir_a_Eliminar()
Descripción:	Se encarga de mandar la acción (Eliminar) hacia el buscador.
Nombre:	Ir_a_Mostrar()
Descripción:	Se encarga de mandar la acción (Mostrar) hacia el buscador.
Nombre:	Ir_a_Renovar()
Descripción:	Se encarga de mandar la acción (Renovar) hacia el buscador.

Nombre: M_Producto	
Tipo de clase: modelo	
Atributo:	**Tipo**
Para cada responsabilidad:	
Nombre:	Insertar($producto)
Descripción:	Inserta en la base de datos un producto que se le pasa por parámetro.
Nombre:	Modificar($producto)
Descripción:	Modifica los datos de un producto que se le pasa por parámetro.
Nombre:	Eliminar($id)
Descripción:	Elimina un producto de la base de datos pasándole el id de ese producto.

Nombre:	Buscar($criterios)
Descripción:	Busca los productos que cumplan con los criterios que se le pasan por parámetro.
Nombre:	Buscar_x_Id($id)
Descripción:	Busca el producto que tenga como id el que se le pasa por parámetro.
Nombre:	Incrementar_Renovacion($id)
Descripción:	Incrementa el número de renovaciones que se le hacen a un producto pasándole el id de dicho producto por parámetro.
Nombre:	Listar_Productos($producto)
Descripción:	Lista los productos que cumplen con las condiciones que se le pasan por parámetros.

Nombre: C_Gestionar_Carga		
Tipo de clase: controladora		
Atributo:		**Tipo:**
Para cada responsabilidad:		
Nombre:	Cargar_Nomencladores ().	
Descripción:	Permite cargar los nomencladores que van a aparecer en las vistas.	
Nombre:	Adicionar()	
Descripción:	Permite adicionar una carga que haya entrado por el puerto.	
Nombre:	Adicionar_Aero()	
Descripción:	Permite adicionar una carga que haya entrado por el aeropuerto.	
Nombre:	Mostrar_V_Notificar()	
Descripción:	Permite mostrar la vista donde se notifica al registro sanitario cuando un producto no esta registrado en el mismo.	
Nombre:	Notificar()	
Descripción:	Permite insertar los datos de la notificación del no registro del producto.	

Nombre:	Mostrar_V_Carga()
Descripción:	Permite mostrar la vista donde se van a insertar los datos de las cargas que entran la país.
Nombre:	Buscar()
Descripción:	Permite buscar los productos que forman parte de la carga.
Nombre:	Add_Producto()
Descripción:	Permite adicionar productos a los datos de una carga que llega al país.
Nombre:	Limpiar_Session()
Descripción:	Permite limpiar la sesión donde se guarda temporalmente los datos de un producto para luego ser insertados.
Nombre:	MBGen_Registrar_Insp()
Descripción:	Función auxiliar que llama al método Mostrar_Buscador_Gen y le pasa una acción (registrar una inspección).
Nombre:	MBGen_Autorizar_Extrac()
Descripción:	Función auxiliar que llama al método Mostrar_Buscador_Gen y le pasa una acción (autorizar una extracción)
Nombre:	MBGen_Ver_Trazab()
Descripción:	Función auxiliar que llama al método Mostrar_Buscador_Gen y le pasa una acción (ver la trazabilidad).
Nombre:	Mostrar_Buscador_Gen ($accion, $controler).
Descripción:	Método que muestra el Buscador de Carga General para realizar la acción indicada.
Nombre:	MBCont_Registrar_Insp()
Descripción:	Función auxiliar que llama al método Mostrar_Buscador_Cont y le pasa una acción (registrar una inspección).
Nombre:	MBCont_Ver_Trazab()
Descripción:	Función auxiliar que llama al método Mostrar_Buscador_Cont y le pasa una acción (ver la trazabilidad).
Nombre:	Mostrar_Buscador_Cont($accion, $controler)
Descripción:	Función que muestra el Buscador de Carga por Contenedor para realizar la acción indicada.
Nombre:	Buscar_Cargas()
Descripción:	Toma los criterios de búsqueda para la carga general y llama al método

	Listar_Cargas.
Nombre:	Buscar_Contenedores()
Descripción:	Toma los criterios de búsqueda para la Carga por Contenedores y llama al método Listar_Cargas
Nombre:	Listar_Cargas($contenedor)
Descripción:	Lista las Cargas que ya tienen Autorización de Extracción.
Nombre:	Mostrar_V_Trazabilidad()
Descripción:	Muestra la vista donde se accede a la trazabilidad de un producto determinado.
Nombre:	Detalles_Inspeccion()
Descripción:	Mediante esta función se accede a todos los datos de una inspección determinada.

Nombre: M_Carga	
Tipo de clase: modelo	
Atributo:	**Tipo:**
Para cada responsabilidad:	
Nombre:	Insertar($carga)
Descripción:	Inserta en la base de datos un objeto de tipo carga que se pasa por parámetro.
Nombre:	Listar_Cargas_Autorizadas($criterios,$per_page,$offset,$contenedor = false)
Descripción:	Lista las cargas que tienen autorizo de extracción.
Nombre:	Listar_Cargas_No_Extraidas($criterios,$per_page,$offset)
Descripción:	Lista las cargas que no tienen autorizo de extracción.
Nombre:	Buscar_x_Id($id, $extraido = false,$tipo_carga = 'general')
Descripción:	Busca una carga utilizando el id como criterio de búsqueda.
Nombre:	Tipo_Carga($id)
Descripción:	Devuelve el tipo de carga pasándole el id como parámetro.
Nombre:	Trazabilidad($id_cp = '', $id_cont = '')
Descripción:	Devuelve todos los datos de un producto, incluye las inspecciones realizadas a este, así como los datos del permiso de extracción.
Nombre:	Get_Certificados($id_carga)
Descripción:	Devuelve los certificados de una carga pasándole el id como parámetro

Nombre:	Get_Inspecciones($id_cp, $id_cont)
Descripción:	Devuelve las inspecciones de una carga pasándole el id como parámetro
Nombre:	Get_Destalles_Insp($id_insp)
Descripción:	Devuelve los detalles de las inspecciones de una carga pasándole el id como parámetro
Nombre:	Get_Incidencia($id_insp)
Descripción:	Devuelve las incidencias de una inspección pasándole el id como parámetro
Nombre:	Get_Certificados_Insp($id_insp)
Descripción:	Devuelve los certificados de las inspecciones de una inspección pasándole el id como parámetro

Nombre: C_Gestionar_Extraccion.		
Tipo de clase: controladora		
Atributo:		**Tipo:**
Para cada responsabilidad:		
Nombre:	Mostrar_V_Extraccion()	
Descripción:	Permite mostrar la vista donde se va a ingresar los datos del permiso de extracción de una carga llegada al país.	
Nombre:	Cargar_Nomencladores ()	
Descripción:	Permite cargar los nomencladores que van a aparecer en las vistas.	
Nombre:	Mostrar_Buscador()	
Descripción:	Permite mostrar la vista que se encarga de buscar las cargas a las cuales se les va a hacer el permiso de extracción.	
Nombre:	Buscar_Carga()	
Descripción:	Permite buscar una carga determinada por varios criterios de búsqueda.	
Nombre:	Listar_Cargas_No_Extraidas()	
Descripción:	Permite listar las cargas que no se le han hecho permiso de extracción.	
Nombre:	Adicionar()	
Descripción:	Permiten hacerle el permiso de extracción a una carga determinada.	
Nombre:	Listar_Permisos()	
Descripción:	Permite listar las cargas a la que se le ha hecho el permiso de extracción.	

Nombre:	Buscar_Carga_Total()
Descripción:	Busca el total de cargas para luego mostrar cierta cantidad por página.
Nombre:	Add_Contenedores()
Descripción:	Permite adicionar los datos de un contenedor para realizar el permiso de extracción.
Nombre:	Listar_Cargas()
Descripción:	Función que permite Listar las cargas
Nombre:	Listar_Incidencias()
Descripción:	Permite listar las incidencias ocurridas durante la inspección realizada a las cargas.

Nombre: M_Permiso_Extracción	
Tipo de clase: modelo	
Atributo:	**Tipo:**
Para cada responsabilidad:	
Nombre:	Insertar($permiso)
Descripción:	Inserta en la base de datos un permiso de extracción hecho a una carga determinada.
Nombre:	Listar($per_page, $offset)
Descripción:	Lista todas las cargas que están en la base de datos.
Nombre:	Get_Puerto($id_carga_prod)
Descripción:	Devuelve el puerto por el cual hizo entrada al país una carga.
Nombre:	Insertar_Contenedor($contenedor)
Descripción:	Se encarga de insertar los contenedores que vienen en una carga por contenedor con sus datos particulares dentro del permiso de extracción hecho a la misma carga.
Nombre:	Trazabilidad()
Descripción:	Devuelve los datos de una carga incluyendo los del permiso de extracción

Nombre: C_Gestionar_IS.	
Tipo de clase: controladora	
Atributo:	**Tipo:**
Para cada responsabilidad:	
Nombre:	Cargar_Nomencladores ()
Descripción:	Permite cargar los nomencladores que van a aparecer en las vistas.
Nombre:	Mostrar_Buscador()
Descripción:	Permite mostrar la vista que se encarga de buscar las cargas a las cuales se les va a hacer la inspección sanitaria.
Nombre:	Mostrar_Buscador_Contenedores()
Descripción:	Permite mostrar la vista que se encarga de buscar las cargas por contenedor a las cuales se les va a hacer la inspección sanitaria.
Nombre:	Mostrar_Vista_Inspeccion()
Descripción:	Permite mostrar la vista donde se va a ingresar los datos de la inspección sanitaria estatal que se le realizan a los productos.
Nombre:	Adicionar()
Descripción:	Permite adicionar una inspección con todas sus especificaciones.
Nombre:	Buscar_Cargas()
Descripción:	Permite buscar las cargas que serán inspeccionadas.
Nombre:	Buscar_Contenedores()
Descripción:	Permite buscar las cargas por contenedor que serán inspeccionadas.
Nombre:	devolverindirectas()
Descripción:	Permite listar las causas indirectas correspondientes a una causa directa en específico.
Nombre:	Listar_Cargas()
Descripción:	Es una función usada por Buscar_Cargas () para listar las cargas que ya tienen permiso de extracción para así pasar a ser inspeccionadas.
Nombre:	devolver_localidades($id_municipio)
Descripción:	Permite listar las localidades de un municipio dado.
Nombre:	devolver_centros($id_centro)
Descripción:	Permite devolver todos los datos de un centro dado mediante el id del mismo.

Nombre:	cant_inspeccion($id_carga_prod)
Descripción:	Permite devolver la cantidad de inspecciones que se le han hecho a una carga determinada.
Nombre:	incrementar_insp($id_carga)
Descripción:	Permite incrementar el número de inspecciones de una carga determinada.
Nombre:	mostrar_especificaciones($id_inspeccion)
Descripción:	Permite devolver todos los datos de una inspección realizada a una carga determinada.
Nombre:	volver_atras()
Descripción:	Método responsable de volver a la página donde se listan la cantidad de inspecciones de una carga cuando se está en la vista en la cual se muestran los datos de una de esas inspecciones realizadas.

Nombre: M_Inspeccion		
Tipo de clase: modelo		
Atributo:		**Tipo:**
Para cada responsabilidad:		
Nombre:	insertar($inspeccion,$tipo_carga)	
Descripción:	Inserta en la base de datos los datos de una inspección hecha a una carga	
Nombre:	dime_liberada($id,$tipo_carga)	
Descripción:	Devuelve si una carga esta liberada o no.	
Nombre:	indirectas($id_directa)	
Descripción:	Devuelve las causas indirectas que pertenecen a una causa directa determinada.	
Nombre:	localidades($id_municipio)	
Descripción:	Devuelve las localidades de un municipio determinado.	
Nombre:	centros($id_centro)	
Descripción:	Devuelve los datos de un centro pasándole como parámetro el id de este.	
Nombre:	cant_inspeccion($id,$tipo_carga)	
Descripción:	Devuelve la cantidad de inspecciones hechas a una carga determinada.	
Nombre:	incrementar_inspeccion($id,$tipo_carga)	

Descripción:	Incrementa la cantidad de inspecciones de una carga.
Nombre:	listar_inspecciones($id,$tipo_carga)
Descripción:	Lista las inspecciones que pertenecen a un tipo de carga determinada.
Nombre:	mostrar_detalles($id_inspeccion,$tipo_carga)
Descripción:	Muestra los detalles de una inspección determinada.
Nombre:	mostrar_certificados($id_inspeccion)
Descripción:	Muestra los certificados de una inspección determinada.
Nombre:	mostrar_incidencia($id_inspeccion)
Descripción:	Muestra la incidencia en caso de que la tenga de una inspección determinada.
Nombre:	Devolver_Datos_Contenedor($id_contenedor)
Descripción:	Devuelve los datos de un contenedor extraído del puerto.

3.3 Descripción de las tablas

A continuación se describen las tablas de la base de datos que gestiona la aplicación para mantener íntegros los datos y gestionar la información almacenada mediante las funcionalidades de la misma, las tablas relacionadas con la seguridad y la configuración se encuentran en el **Anexo 3**(***Tablas de la base de datos relacionadas con la seguridad y configuración***) y el diseño de la base de datos se encuentra en el **Anexo 1**(***Diseño de la base de datos***).

Nombre: producto		
Descripción: Esta tabla almacena los datos generales de un producto alimenticio de importación.		
Atributo	**Tipo**	**Descripción**
id.	Integer.	Es el identificador para los datos de esta tabla.
id_pais.	Integer.	Llave extranjera que representa la relación con la tabla país.
id_grupo.	Integer.	Llave extranjera que representa la relación con la tabla grupo.
id_empresa.	Integer.	Llave extranjera que representa la relación con la tabla empresa.
no_licencia.	Varchar.	Este atributo es el número de licencia de cada producto.
renovacion	Integer.	Enumera la cantidad de veces que un producto ha sido renovado

fecha_registro.	DateTime.	Almacena la fecha de registro del producto.
fecha_renovacion.	DateTime.	Almacena la fecha de renovación de un producto.
no_orden	Integer.	Es el número de orden de cada producto.
nombre_producto	Varchar	Es el nombre del producto.
marca	Varchar	Es la marca de cada producto.
fabricante	Varchar	Es el nombre del fabricante del producto.
eliminado	Tinyint	Identifica cuando un producto está eliminado mediante el valor que tome (true o false).

Nombre: indicador_fisico_quimico		
Descripción: Esta tabla almacena los indicadores físicos químicos de los productos.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
id_producto	Integer	Llave extranjera que representa la relación con la tabla producto
cloruro	Varchar	Es el valor de cloruro que tiene el producto.
acidez	Integer	Es el valor de acidez que tiene el producto.
ph	Integer	Es el valor de ph que tiene el producto.
i_peroxido	Integer	Es el valor de i_peroxido que tiene el producto.
humedad	float	Es el valor de humedad que tiene el producto.
co2	float	Es el valor de co2 que tiene el producto.
cenizas	Varchar	Es el valor de cenizas que tiene el producto.
organolepticos	Varchar	Es el valor de organolépticos que tiene el producto.
hermeticidad	Varchar	Es el valor de hermeticidad que tiene el producto.
rancidez_cualitativa	Varchar	Es el valor de rancidez cualitativa que tiene el producto.
rancidez_cuantitativa	Varchar	Es el valor de rancidez cuantitativa que tiene el producto.
solidos_solubles	Varchar	Es el valor de sólidos solubles que tiene el producto.
solidos_no_grasos	Varchar	Es el valor de sólidos no grasos que tiene el producto.
i_yodo	float	Es el valor de i_yodo que tiene el producto.
i_saponificacion	Integer	Es el valor de i_saponificacion que tiene el producto.

punto_fusion	Varchar	Es el valor de punto fusión que tiene el producto.
peso_especifico	float	Es el valor de peso específico que tiene el producto.
metanol	Varchar	Es el valor de metanol que tiene el producto.
grado_alcohol	float	Es el valor de grado alcohol que tiene el producto.
alcohol_superficial	Varchar	Es el valor de alcohol superficial que tiene el producto.
i_refraccion	Integer	Es el valor de i_refraccion que tiene el producto.
volatilidad	Varchar	Es el valor de volatilidad que tiene el producto.
temp_solidificacion	float	Es el valor de temperatura de solidificación que tiene el producto.
alcali_libre	Varchar	Es el valor de álcali libre que tiene el producto.
mono_trigliceridos	Varchar	Es el valor de mono triglicéridos que tiene el producto.
acidez_volatil	float	Es el valor de acidez volátil que tiene el producto.
acidez total	float	Es el valor de acidez total que tiene el producto.
masa_neta	float	Es el valor de masa neta que tiene el producto.
polarizacion	Varchar	Es el valor de polarización que tiene el producto.
color	Varchar	Es el color característico de dicho producto.
glucosa	Varchar	Es el valor de glucosa que tiene el producto.
sacarosa	Varchar	Es el valor de sacarosa que tiene el producto.
azucares_reductores	Varchar	Es el valor de azucares reductores que tiene el producto.
conductividad	Varchar	Es el valor de la conductividad que tiene el producto.
materia_insaponificable	Varchar	Es el valor de materia insaponificable que tiene el producto.
centro_termico	Varchar	Es el valor del centro térmico que tiene el producto.
nvb	Varchar	Es el valor del nvb que tiene el producto.

Nombre: indicador_microbiologico		
Descripción: Esta tabla almacena los indicadores microbiológicos de los productos.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
id_producto	Integer	Llave extranjera que representa la relación con la tabla producto.
i_monocxitogenes	Integer	Es el valor de i_monocxitogenes que tiene el producto.
coliformes_totales	Integer	Es el valor de coliformes_totales que tiene el producto.

coliformes_fecales	Integer	Es el Conteo de Coliformes Fecales del producto.
hongos_levadura	Integer	Es el Conteo de Hongos levadura del producto.
conteo_total	Integer	Es el Conteo de hongos que tiene el producto.
s_aureus	Varchar	Es el valor de S. Aureus del producto.
salmonella	Varchar	Es el valor de Salmonella del producto.
prueba_esterilidad	Varchar	Es el valor de la Prueba de Esterilidad del producto.
p_aeruginosa	Varchar	Es el valor de la Prueba de Pseudomononas Aeruginosa del producto.
e_coli	Integer	Es el valor del indicador e_coli del producto.
p_aeruginosa	Varchar	Es el valor de la prueba aeruginosa que se le hace al producto
b_cereus	Integer	Es el valor del indicador b_cereus del producto.
camara_howard	Varchar	Es el valor de la prueba Cámara Howard que se le hace al producto.
c_sulfito_reductores	Integer	Es el valor del conteo de sulfitos reductores del producto.
v_cholerae	Integer	Es el valor del indicador v cholerae del producto.
v_hemolyticus	Integer	Es el valor del indicador v hemolyticus del producto.

Nombre: indicador_toxicologico		
Descripción: Esta tabla almacena los indicadores toxicológicos de los productos.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
id_producto	Integer	Llave extranjera que representa la relación con la tabla producto.
plomo	Varchar	Es el valor del indicador plomo que tiene el producto.
estano	Varchar	Es el valor del indicador estaño que tiene el producto.
mercurio	Varchar	Es el valor del indicador mercurio que tiene el producto
hierro	Varchar	Es el valor del indicador hierro que tiene el producto
histamina	Varchar	Es el valor del indicador histamina que tiene el producto
aflatoxina	Varchar	Es el valor del indicador aflatoxina que tiene el producto
vomitoxina	Varchar	Es el valor del indicador vomitoxina que tiene el producto
adictiva	Varchar	Es el valor del indicador adictiva que tiene el producto

antibioticos	Varchar	Es el valor del indicador antibióticos que tiene el producto
cadmio	Varchar	Es el valor del indicador cadmio que tiene el producto
arsenico	Varchar	Es el valor del indicador arsénico que tiene el producto
patulina	Varchar	Es el valor del indicador patulina que tiene el producto
radionucleos	Varchar	Es el valor del indicador radio núcleos que tiene el producto
plaguicida	Varchar	Es el valor del indicador plaguicida que tiene el producto

Nombre: info_nutricional		
Descripción: Esta tabla almacena la información nutricional de los productos.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
id_producto	Integer	Llave extranjera que representa la relación con la tabla producto.
proteinas	float	Es el valor proteínico que tiene el producto.
grasas	float	Es el valor de grasas que tiene el producto.
carbohidratos	float	Es el valor de carbohidratos que tiene el producto
vitaminas	float	Es el valor de las vitaminas que tiene el producto
minerales	float	Es el valor de los minerales que tiene el producto

Nombre: carga		
Descripción: Esta tabla almacena la información relacionada con una carga		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
no_contrato	Integer	Se trata del número de contrato de una carga en particular.
nombre_nave	Varchar	Es el nombre que identifica a una nave.
manifiesto	Varchar	Es el número de manifiesto de una carga.
fecha_incicio_descarga	Date	Es la fecha de inicio de descarga de una determinada carga.
fecha_fin_descarga	Date	Es la fecha del fin de descarga de una determinada

		carga.
fecha_inicio_carga_origen	Date	Se trata de la fecha del inicio de la carga en el origen.
fecha_fin_carga_origen	Date	Almacena la fecha del fin de la carga en el origen.
fecha_arribo	Date	Almacena la fecha de arribo de la carga a Cuba.
fecha_embarque	Date	Es la fecha de embarque de una carga determinada.
id_fumigacion	Integer	Representa la relación con la tabla fumigación.

Nombre: carga_aeropuerto		
Descripción: Esta tabla almacena la información relacionada con una carga que arriba por un aeropuerto.		
Atributo	**Tipo**	**Descripción**
id_carga_aeropuerto	Integer	Es el identificador para los datos de esta tabla.
no_factura	Integer	Se trata del número de factura de una carga en particular que arriba por un aeropuerto.
id_aeropuerto	Integer	Representa la relación con la tabla n_aeropuerto y almacena el id del aeropuerto con el que esta relacionada una carga.
vuelo	Varchar	Es el número de vuelo de una carga.
guía_aerea	Varchar	Se trata de la guía aérea de un vuelo que arriba con una carga.

Nombre: carga_puerto		
Descripción: Esta tabla almacena la información relacionada con una carga que arriba por un puerto.		
Atributo	**Tipo**	**Descripción**
id_carga_puerto	Integer	Es el identificador para los datos de esta tabla.
ultimo_puerto_visitado	Varchar	Se trata del último puerto que visitó el barco que viene con la carga.
id_terminal	Integer	Representa la relación con la tabla n_terminal.

Nombre: carga_general		
Descripción: Esta tabla almacena la información relacionada con una carga general.		
Atributo	**Tipo**	**Descripción**
id_carga_general	Varchar	Es el identificador para los datos de esta tabla.
carga_anterior	Varchar	Se trata de la carga anterior que estuvo en el barco antes de la actual.
fecha_limpieza	Date	Es la fecha en que se limpió la bodega donde vino una carga determinada.
ventilacion_carga	Varchar	Es la descripción de la ventilación de la carga (buena o mala).
humedad_bodega	float	Es el valor de la humedad de la bodega donde vino la carga.
temperatura_promedio	float	Se trata del valor de la temperatura promedio de la bodega donde vino la carga.
incidencias_travesia	text	Se trata de la descripción de alguna incidencia ocurrida durante la travesía de la embarcación.

Nombre: carga_contenedor		
Descripción: Esta tabla almacena la información relacionada con una carga general.		
Atributo	**Tipo**	**Descripción**
id_carga_contenedor	Integer	Es el identificador para los datos de esta tabla.
temperatura	float	Es la temperatura de los contenedores donde viene la carga.
tipo_contenedor	Integer	Se trata de la clasificación de los contenedores (20 o 40) pies.

Nombre: contenedores_extraidos		
Descripción: Esta tabla almacena los datos de cada contenedor que tenga permiso de extracción.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
id_permiso	Integer	Representa la relación con la tabla permiso_extraccion y se almacena el id del permiso de extracción con el que

		esta relacionado dicho contenedor.
no_contenedor	Varchar	Almacena el número del contenedor extraído.
id_centro	Integer	Representa la relación con la tabla n_centro y se almacena el id del centro que tiene como destino el contenedor.
retenido	Tinyint	Identifica si esta retenido o no el contenedor.
id_puerto	Integer	Representa la relación con la tabla n_puerto y almacena el id del puerto que es destino de algún contenedor.
retenido_por	Varchar	Se trata de la causa por la que se retuvo el contenedor.
cant_inspecciones	Integer	Almacena la cantidad de inspecciones que se le ha hecho a una carga por contenedor.
liberado	Tinyint	Señala si un producto está o no liberado.

Nombre: permiso_extracción		
Descripción: Esta tabla almacena los datos del permiso de extracción que se le realiza a toda carga que arriba a Cuba.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
funcionario	Varchar	Es el nombre del funcionario que esta realizando el permiso de extracción
fecha_extraccion	Date	Almacena la fecha en que la carga se extrae.
cantidad	float	Recoge la cantidad que se va a extraer.
id_unidad	Integer	Representa la relación con la tabla n_unidad_medida y almacena el id de la unidad de medida que tiene la carga.
interesado		Almacena el nombre de la persona que tiene interés en darle permiso de extracción a la carga.
ci	Varchar	Guarda el número del carnet de identidad de la persona interesada en hacerle el permiso de extracción a una carga.
id_carga_prod	Integer	Representa la relación con la tabla tr_carga_producto y almacena el id de una carga de un producto determinado a la cual se le realiza el permiso de extracción.
fecha_solicitud	Date	Almacena la fecha en que se solicitó el permiso de

		extracción de la carga.
id_centro	Integer	Representa la relación con la tabla n_centro y almacena el id del centro que tiene como destino la carga.

Nombre: inspección		
Descripción: Esta tabla almacena los datos de las inspecciones que se le realizan a las cargas de productos alimenticios.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
unidad_ejecutora	Varchar	Es el nombre de la unidad ejecutora de la inspección.
giro	Varchar	Almacena el giro de la carga que es inspeccionada.
observaciones	text	Almacena las observaciones durante la inspección.
fecha_inspeccion		Este campo guarda la fecha en que se realizó la inspección.
no_contenedor	Integer	Almacena el número de contenedor al que se le hizo la inspección en caso de que la carga sea por contenedor.
cantidad_peso_neto	Integer	En caso de que la carga sea general guarda la cantidad que se inspeccionó.
fecha_produccion	Date	Almacena la fecha de producción del producto que viene en la carga inspeccionada.
fecha_vencimiento	Date	Almacena la fecha de vencimiento del producto que viene en la carga inspeccionada.
funcionario	text	Se trata del funcionario que realizó la inspección.
propietario	text	Es el propietario de la carga que se inspeccionó.
cant_liberada	float	Representa la cantidad que se liberó durante la inspección.
id_centro	Integer	Representa la relación con la tabla n_centro y almacena el id de el centro que va a estar relacionado con la inspección.
cant_decomisada	Integer	Se trata de la cantidad decomisada durante la inspección sanitaria.
id_carga_prod	Integer	Representa la relación con la tabla tr_carga_producto y

		almacena el id de una carga de un producto a la que se le realizó la inspección.
id_contenedor	Integer	Representa la relación con la tabla contenedores_extraidos y almacena el id del contenedor que se le realizó la inspección.

Nombre: incidencia		
Descripción: Esta tabla almacena los datos de las incidencias ocurridas durante las inspecciones sanitarias		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
id_inspeccion	Integer	Representa la relación con la tabla inspección y almacena el id de la inspección donde se identificó la incidencia.
accion_tomada	Varchar	Almacena la acción tomada durante una incidencia.
cant_implicada	Integer	Es la cantidad implicada en una incidencia en caso que la carga inspeccionada fuese general.
conflicto	Tinyint	Mediante este atributo se identifica la incidencia ocurrida como conflicto sanitario.
observaciones	text	Representa las observaciones de la incidencia.

Nombre: pendiente_registro		
Descripción: Esta tabla almacena los datos de productos pendientes de registro sanitario.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
id_causa	Integer	Representa la relación con la tabla n_causa_no_registro y almacena el id de la causa de no registro por la cual el producto está pendiente de no registro.
id_conducta	Integer	Representa la relación con la tabla n_conducta_seguir y almacena el id de la conducta a seguir ante un producto pendiente de registro.
nombre_producto	Varchar	Almacena el nombre del producto que esta pendiente de registro.

marca		Es la marca de un producto.
fabricante	Varchar	Es el fabricante del producto.
id_empresa	Integer	Representa la relación con la tabla n_empresa_importadora y almacena el id de la empresa importadora del producto.

Conclusiones

En este capítulo se llevó a cabo la descripción y el análisis de la solución propuesta, donde se encuentra el análisis de componentes o módulos ya existentes que son rehusados proporcionándole a la aplicación servicios que le son de utilidad para su correcto funcionamiento; la descripción de las tablas de la base de datos, lo cual permite conocer con exactitud los mecanismos de almacenamiento utilizados; así como la descripción de las clases utilizadas en la implementación y sus principales funcionalidades que responden a las necesidades del cliente.

CONCLUSIONES

Al culminar el presente trabajo se arribó a las siguientes conclusiones:

- ✓ Se valoraron las tecnologías a utilizar para el desarrollo de la aplicación.
- ✓ Se elaboró el Modelo de Implementación.
- ✓ Se elaboraron los artefactos necesarios para describir la Base de Datos.
- ✓ Se diseñó la Base de Datos.
- ✓ Se integró la aplicación con los componentes definidos del Sistema de Información para la Salud (SiSalud).

Con el desarrollo de este trabajo se le dio cumplimiento al objetivo general propuesto, implementándose una aplicación web que facilita la gestión de la información relacionada con la higiene de los alimentos de importación en Cuba.

RECOMENDACIONES

Con el objetivo de lograr mejoras en el funcionamiento de la aplicación se recomienda:

- ✓ Utilizar las librerías: Yahoo User Interface (YUI), incluye el manejo de Ajax y le proporciona una interface más amigable para el usuario.
- ✓ Definir un conjunto de reportes estadísticos que puedan imprimirse y así facilitar aun más el manejo y organización de la información.
- ✓ Extender el alcance de la aplicación a cualquier producto que sea importado por el país, no solo limitarse a los alimentos.

REFERENCIAS BIBLIOGRÁFICAS:

1. **Luján Mora, Sergio.** Publicaciones. [En línea] [Citado el: 12 de febrero de 2008.] http://gplsi.dlsi.ua.es/almacenes/publicaciones.php?id=1&idpub=-1.

2. Aplicación web. [En línea] [Citado el: 12 de diciembre de 2007.] http://meteo.ieec.uned.es/www_Usumeteo2/Memoria/Capitulo3.pdf.

3. Ídem a la referencia 2. [En línea]

4. **Vegas, Jesús.** Introducción a las aplicaciones web. [En línea] [Citado el: 28 de noviembre de 2007.] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node11.html.

5. **Delgado, Héctor.** INFORMHECTOR. [En línea] [Citado el: 6 de febero de 2008.] http://informhector.blogspot.com/.

6. **Troya, José M. y Vallecillo, Antonio.** Desarrollo de Software Basado en Componentes. [En línea] [Citado el: 12 de Enero de 2008.] http://www.lcc.uma.es/~av/Docencia/Doctorado/tema1.pdf.

7. Terminología. [En línea] [Citado el: 16 de Abril de 2008.] http://wai.com.ve/menu1/termino.html#Javascript.

8. Crea tu Web. [En línea] [Citado el: 11 de Enero de 2008.] http://www.creartuweb.org/index.php.

9. Ídem a la referencia 7. [En línea]

10. El espacio de la Iguana. [En línea] [Citado el: Enero de 12 de 2008.] http://bjuarezorieta.blogspot.es/1208205840/.

11. Ídem a referencia 8. [Online]

12. webestilo. [En línea] [Citado el: 3 de Mayo de 2008.] http://www.webestilo.com/mysql/intro.phtml.

13. Entersoft. [En línea] [Citado el: 12 de Enero de 2008.] http://www.entersoft.com.mx/Sitio/plataformaTecnologica.asp.

14. Lazos. [En línea] [Citado el: 2 de marzo de 2008.] http://titanio.lazos.cl/lazos/index.php?id=31.

15. Técnico en Prácticas. [En línea] [Citado el: 5 de marzo de 2008.] http://montse-tecnicoenpracticas.blogspot.com/2008/02/redes-ii.html.

16. ProfesionalHosting. [Online] [Cited: Mayo 20, 2008.] http://www.profesionalhosting.com/servidores-dedicados/definicion/servidor-httpdapache-70.html.

17. **Cornejo Román, Richard E.** GNU/Linux como SO para servidor. [Online] [Cited: Junio 2, 2008.]

18. Intro Ingeniería Softaware. [Online] [Cited: Mayo 29, 2008.]

19. EL Codigo K. [Online] [Cited: Junio 4, 2008.] http://elcodigok.blogspot.com.

20. Ídem a Referencia 19.

21. GenBiz. [Online] [Cited: Junio 4, 2008.] http://www.genbiz.cl.

22. La Plataforma Microsoft . NET. [Online] [Cited: Junio 5, 2008.] http://www.lcc.uma.es/~pastrana/EP/trabajos/45.pdf.

23. Essential Studios. [Online] [Cited: Junio 3, 2008.] http://essential-studios.net/.

24. *Ídem a Referencia 19.*

25. Artículos web. [Online] [Cited: Junio 7, 2008.] http://www.articuloweb.com/category.php?cat_id=43&start=2.

26. Desarrollo Web. [Online] [Cited: Junio 7, 2008.] http://www.desarrolloweb.com/articulos/1178.php.

27. W3C. [Online] [Cited: Junio 7, 2008.] http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo.

28. Programación Web. [Online] [Cited: Junio 7, 2008.] http://www.programacionweb.net/articulos/articulo/?num=505.

29. Proactiva-Calidad. [Online] [Cited: Junio 8, 2008.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

30. msdn. [Online] [Cited: Junio 8, 2008.] http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3985/default.aspx.

31. el guille. [Online] [Cited: Junio 12, 2008.]

http://www.elguille.info/colabora/NET2005/julio_casal_Componentes.htm.

32. Scribd. [Online] [Cited: Junio 9, 2008.] http://www.scribd.com/doc/272974/Cliente-servidor.

33. MasterMagazine. [Online] http://www.mastermagazine.info/termino/6498.php .

34. [Online] [Cited: Junio 9, 2008.] http://www.inei.gob.pe/biblioineipub/bancopub/inf/lib5038/ven1.HTM.

35. Universidad Técnica de Loja. [Online] [Cited: Junio 9, 2008.] http://www.utpl.edu.ec/blog/zamora/2008/02/13/arquitectura-por-capas/.

36. Blog. [Online] [Cited: Junio 10, 2008.] http://www.utpl.edu.ec/blog/jlbautista/2008/02/04/10/.

37. *Ídem a Referencia 36.*

38. *Ídem a Referencia 36.*

39. *Ídem a Referencia 36.*

40. [Online] [Cited: Junio 10, 2008.] http://www-gris.det.uvigo.es/~avilas/UML/node49.html .

BIBLIOGRAFÍA

- ✓ Portal de la salud en Cuba. [Online] http://www.sld.cu/sistema_de_salud/metodologica/epidemiologia.html#INTERNACIONAL.
- ✓ REGLAMENTO DE LAS DISPOSICIONES E INFRACCIONES SOBRE CONTROL SANITARIO INTERNACIONAL. [Online] http://www.medioambiente.cu/legislacionE/decretos/D-104.htm.
- ✓ Portal de Medio Ambiente en Cuba. [Online] http://www.medioambiente.cu/.
- ✓ Informática en la salud pública cubana. [Online] http://bvs.sld.cu/revistas/spu/vol32_3_06/spu15306.htm.
- ✓ *Manual de Regulaciones establecidas para el Registro Sanitario,3ra Versión.*
- ✓ *Manual de Indicadores Sanitarios Empleados en la Evaluación de Alimentos, Cosméticos, Artículos de Aseo , Juguetes y Tecnologías Ambientales.* Ciudad de la Habana : s.n., 2004.
- ✓ La Salud en Cuba. [Online] http://scielo.sld.cu.
- ✓ **Pressman, Roger S.** *Íngenieria del software 5ta edición.*
- ✓ **Larman, Craig.** *UML y Patrones.*
- ✓ **Jacobson, Ivar, Booch, Grady and Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.*
- ✓ Portal Universidad de Alicante. [Online] http://gplsi.dlsi.ua.es/almacenes/publicaciones.php?id=1&idpub=-1..
- ✓ Portal de la Universidad de Malaga. [Online] http://www.lcc.uma.es/~av/Docencia/Doctorado/tema1.pdf.
- ✓ portal El Espacio de la Iguana. [Online] http://bjuarezorieta.blogspot.es/1208205840/.
- ✓ WebEstilo. [Online] http://www.webestilo.com/mysql/intro.phtml.
- ✓ EnterSoft. [Online] http://www.entersoft.com.mx/Sitio/plataformaTecnologica.asp.
- ✓ Lazos. [Online] http://titanio.lazos.cl/lazos/index.php?id=31.
- ✓ Técnico en Prácticas. [Online] http://montse-tecnicoenpracticas.blogspot.com/2008/02/redes-ii.html.
- ✓ Profesional Hosting. [Online] http://www.profesionalhosting.com/servidores-dedicados/definicion/servidor-httpdapache-70.html.

- ✓ El codigo K. [Online] http://elcodigok.blogspot.com/.
- ✓ GenBiz. [Online] http://www.puntogen.cl/genbiz.html.
- ✓ Essential Studios. [Online] http://essential-studios.net/.
- ✓ Portal Desarrollo Web. [Online] http://www.desarrolloweb.com/articulos/1178.php.
- ✓ Programación Web. [Online] http://www.programacionweb.net/articulos/articulo/?num=505.
- ✓ Portal de Microsoft. [Online] http://microsoft.com.
- ✓ Master Magazine. [Online] http://www.mastermagazine.info/termino/6498.php.
- ✓ Portal de la Universidad Técnica de Loja. [Online] http://www.utpl.edu.ec/blog/zamora/2008/02/13/arquitectura-por-capas/.
- ✓ Blog. [Online] http://www.utpl.edu.ec/blog/jlbautista/2008/02/04/10/.
- ✓ Portal de las Estadisticas de Salud de La República de Cuba. [Online] http://www.dne.sld.cu/minsap/mision.htm.
- ✓ Manual de php. [Online] http://www.php.net/manual/es.
- ✓ Tutorial de CodeIgniter. [Online] http://codeigniter.com/tutorials.
- ✓ **Gilfillan, Ian.** *La biblia de MySQL.*
- ✓ **Vilalta, Josep.** *UML Guia Visual.*
- ✓ saludambiental.net. [Online] http://www.saludambiental.net.
- ✓ Higiene y Seguridad Alimentaria. [Online] http://higiene.unex.es/.
- ✓ buenaSalud. [Online] http://www.buenasalud.com/lib/ShowDoc.cfm?LibDocID=3466&ReturnCatID=5.
- ✓ Nutrición Especializada. [Online] http://www.nutricionespecializada.com/higiene.html.
- ✓ Organización de las Naciones Unidas para la Agricultura y la Alimentación. [Online] http://www.fao.org/index_ES.htm.
- ✓ Organización mundial de la Salud. [Online] http://www.who.int/es/.
- ✓ Programa nacional de Control Sanitario Internacional. [Online] http://aps.sld.cu/bvs/materiales/programa/csi/csi.pdf.

- ✓ Ley No.41 sobre la salud pública cubana. [Online] http://www.trabajadores.cu/materiales_especiales/suplementos/mundo-laboral/legislacion-laboral/ley-no-41-sobre-la-salud-publica/?searchterm=priorizada.
- ✓ Revista Habanera de Ciencias Médicas. [Online] http://www.ucmh.sld.cu/rhab/editorial_rev10.htm.

ANEXOS

Anexo 1. Diseño de la Base de Datos.

Diagrama Entidad-Relación (Registro Sanitario).

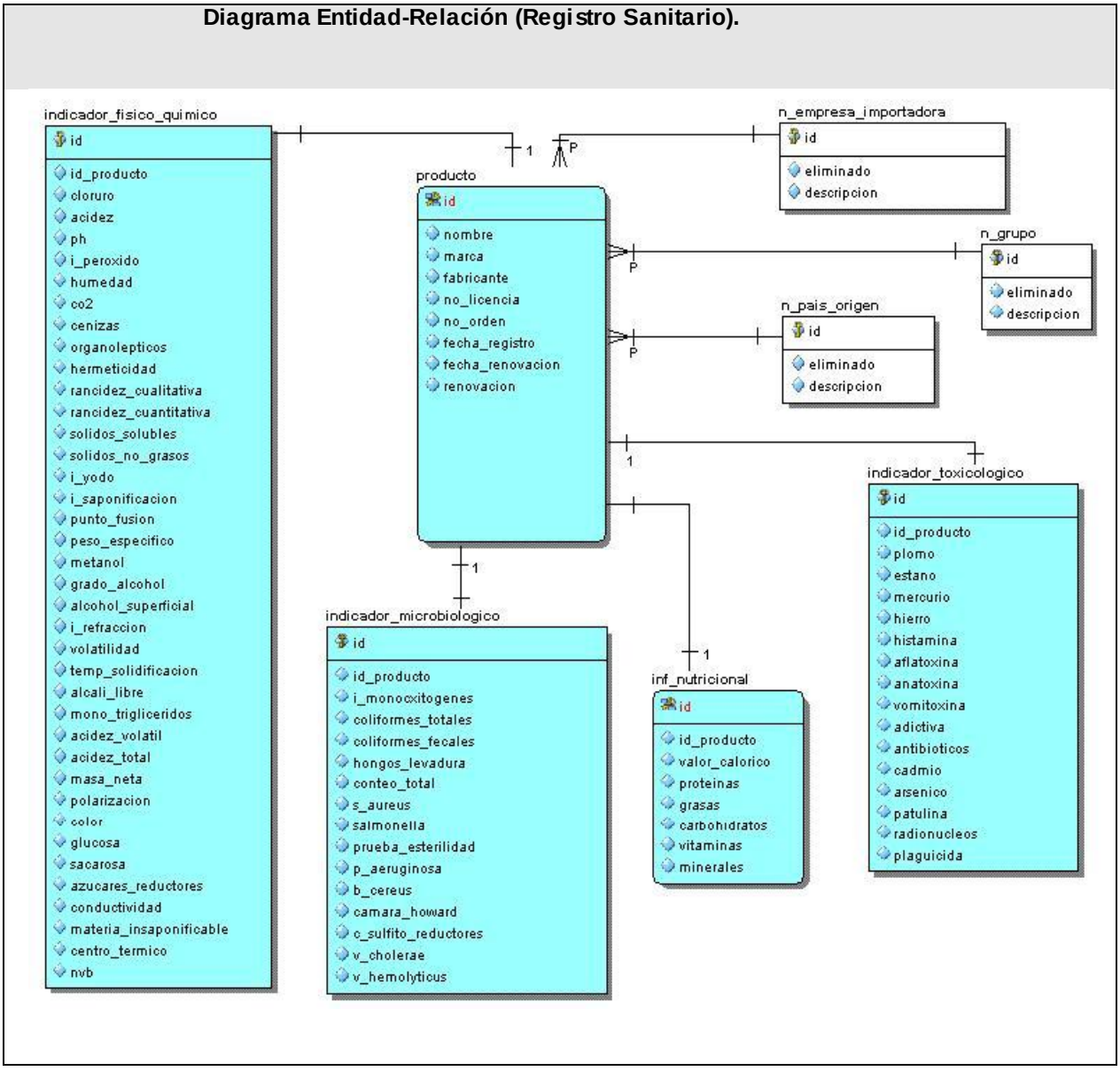


Diagrama Entidad-Relación (Datos de la carga).

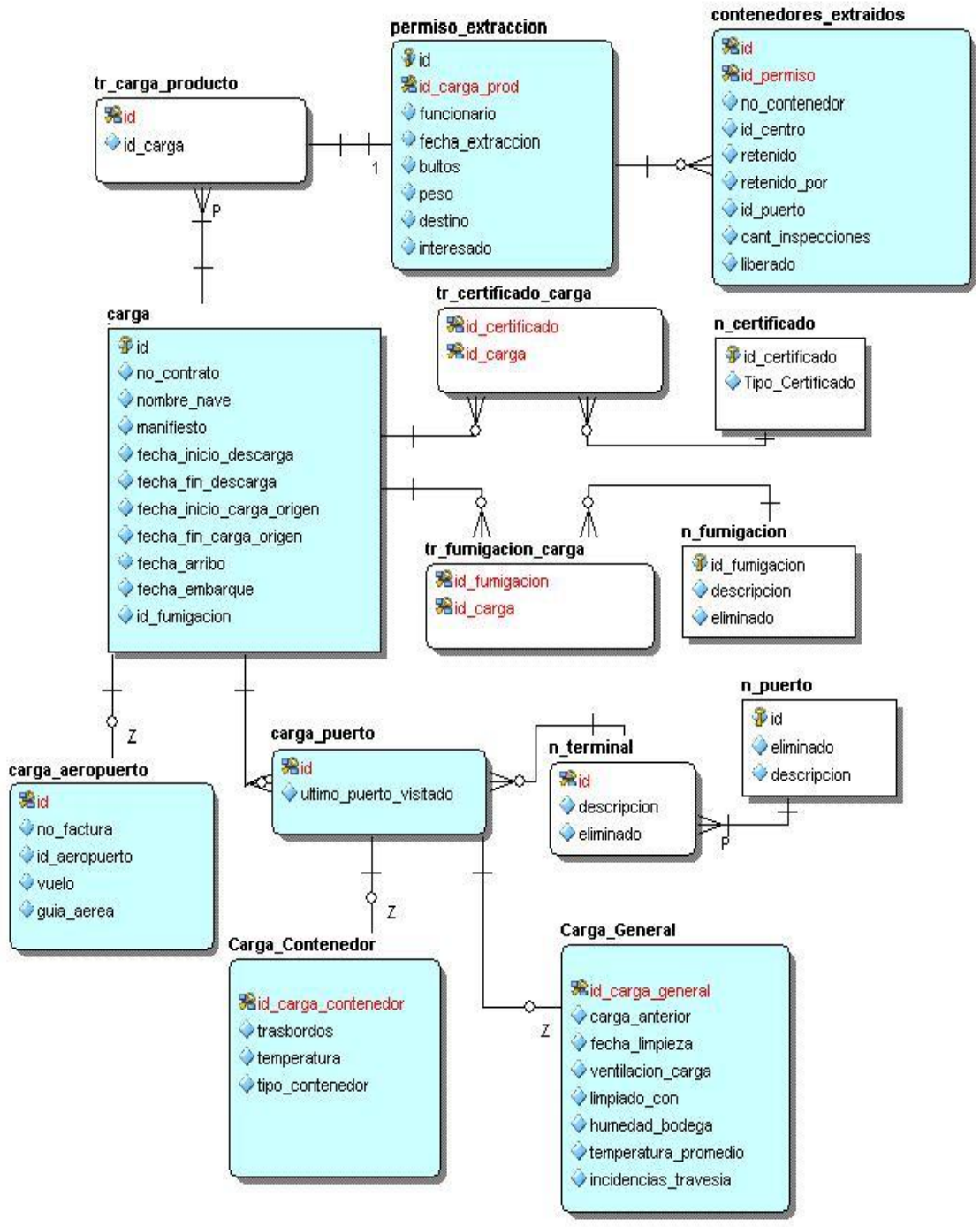


Diagrama Entidad-Relación (Datos de la Inspección Sanitaria).

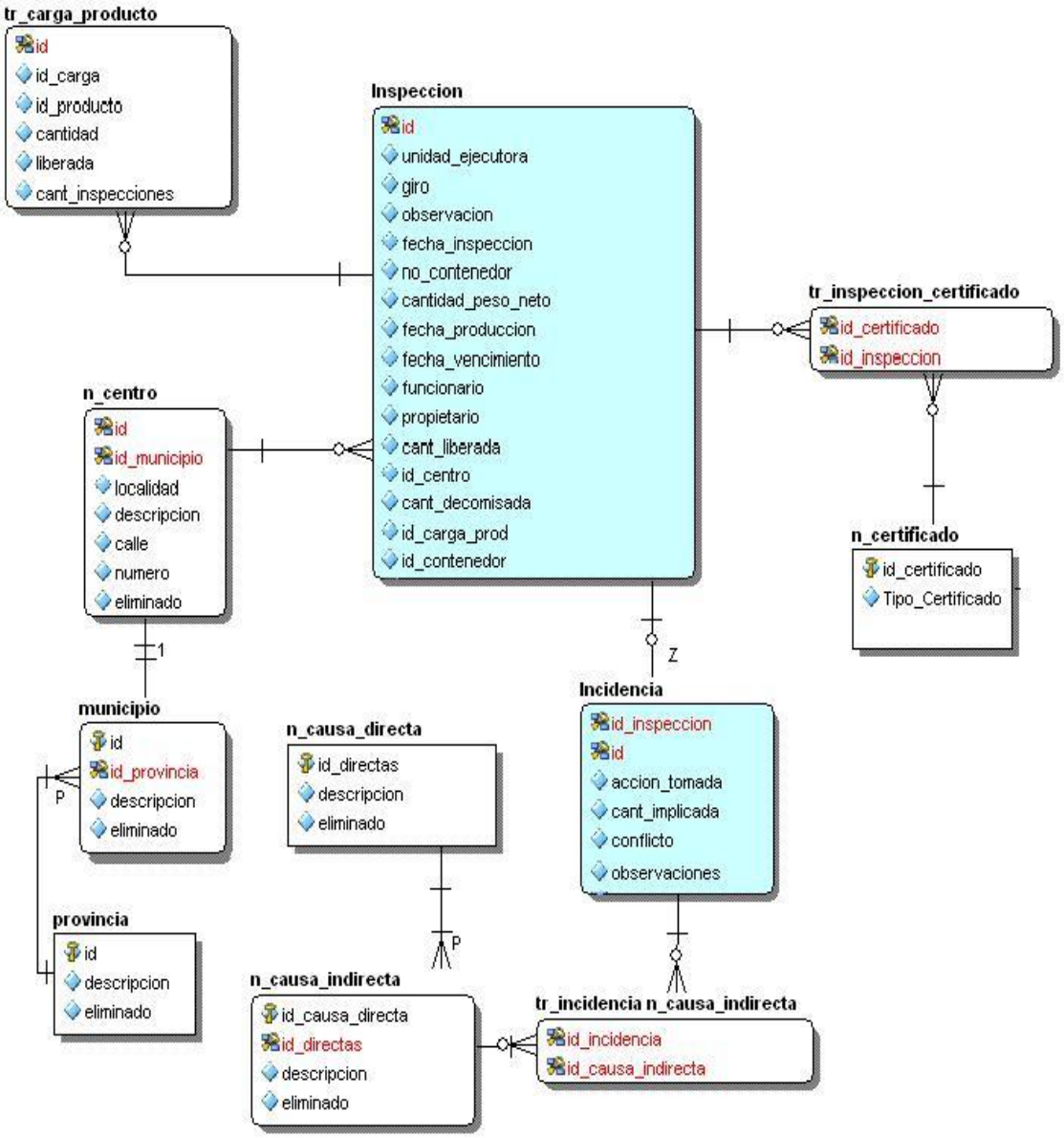
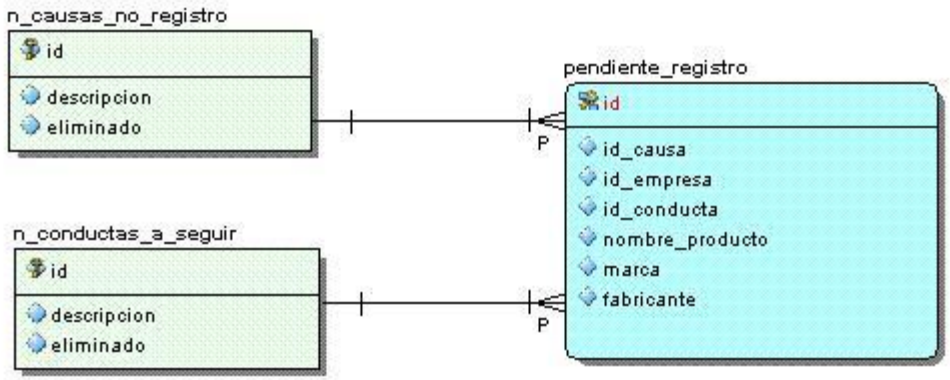


Diagrama Entidad-Relación (Pendientes de Registro).



Anexo 2. Descripción de clases de configuración y seguridad

Nombre: C_Usuario	
Tipo de clase: controladora	
Atributo:	**Tipo**
Para cada responsabilidad:	
Nombre:	AutenticarSAAA()
Descripción:	Permite la autenticación de un usuario a través del SAAA.
Nombre:	Login()
Descripción:	Permite la autenticación de un usuario.
Nombre:	ExitConfirm()
Descripción:	Permite confirmar la salida de la aplicación.
Nombre:	Salir()
Descripción:	Permite salir de la aplicación.

Nombre: M_Usuarios	
Tipo de clase: modelo	
Atributo:	**Tipo**
Para cada responsabilidad:	
Nombre:	HaveAutorization($rol,$link)
Descripción:	Dado un id de rol y un vínculo devuelve si el rol puede invocar al vínculo
Nombre:	GetLinks4User($rol)
Descripción:	Devuelve dado un rol todos los vínculos a los que tiene acceso para mostrarlos en el menú.
Nombre:	GetUserData($username)
Descripción:	Devuelve los datos de un usuario
Nombre:	Buscar_Usuario($user)
Descripción:	Este comprueba que el usuario exista en la BD.
Nombre:	Get_Password($user)
Descripción:	Comprueba que el password exista en la BD para un usuario determinado.
Nombre:	Tiene_Acceso($user)
Descripción:	Selecciona el rol de un usuario existente en la BD.
Nombre:	Nombre($user)
Descripción:	Devuelve el nombre del usuario existente en la BD.
Nombre:	Descripcion_Rol($user)
Descripción:	Devuelve la descripción de un rol determinado.
Nombre:	UpdateUsers($rol,$pass,$repass,$segment)

Descripción:	Actualiza el password y el rol de un usuario del sistema.
Nombre:	CreateUsers($user,$rol,$pass,$repass,$segment)
Descripción:	Actualiza el usuario, el password y el rol en la tabla de la BD.

Nombre: C_Configuracion	
Tipo de clase: controladora	
Atributo:	**Tipo**
Para cada responsabilidad:	
Nombre:	V_Nomencladores()
Descripción:	Permite mostrar las vistas para gestionar los nomencladores.
Nombre:	Adicionar()
Descripción:	Permite adicionar un nomenclador.
Nombre:	Eliminar()
Descripción:	Permite eliminar un nomenclador.
Nombre:	Editar()
Descripción:	Permite editar un nomenclador.
Nombre:	GestionarTabla($tabla_bd, $per_page, $controller, $view)
Descripción:	Permite mostrar un listado de nomencladores en una vista.
Nombre:	GestionarTablaHija($tabla_bd, $per_page = 20, $controller, $view,$id_padre, $descripcion)
Descripción:	Permite mostrar un listado de nomencladores que son dependientes de otros en una vista.
Nombre:	Cargar_Nomencladores()
Descripción:	Permite cargar los nomencladores respectivos para insertar el nomenclador centro.
Nombre:	Nomenc_Relacionados ($id_padre, $desc_id, $tabla_hijo).
Descripción:	Permite obtener un listado de nomencladores que están relacionados.

Nombre: M_Configuracion	
Tipo de clase: modelo	
Atributo:	**Tipo**
Para cada responsabilidad:	
Nombre:	Insertar($nomenclador,$tabla)
Descripción:	Inserta un nomenclador determinado en la tabla que le corresponde
Nombre:	Modificar($nomenclador,$tabla)
Descripción:	Modifica la descripción de un nomenclador determinado
Nombre:	Eliminar($id, $tabla)
Descripción:	Elimina un nomenclador de la tabla pasándole el id del mismo por parámetro
Nombre:	Cargar_Nomenclador($tabla)
Descripción:	Lista todos los elementos de la tabla que se le pasa por parámetros.
Nombre:	Buscar_x_Id($id, $tabla)
Descripción:	Busca y devuelve el elemento en la tabla que se pasa por parámetro
Nombre:	Nomencladores_Relacionados($id_padre, $desc_id, $tabla_hijo)
Descripción:	Dado un nomenclador devuelve todos los que son dependientes de este.
Nombre:	Activar($id, $tabla)
Descripción:	Activa los nomencladores que están desactivados, es decir modifica el campo eliminado con el valor '' true''.
Nombre:	obtener_centros()
Descripción:	Devuelve todos los centros que están insertados en la base de datos.
Nombre:	Obtener_Direccion($id_centro)
Descripción:	Dado un centro devuelve la dirección del mismo.
Nombre:	comprobar_existencia($tabla,$Descripcion)
Descripción:	Comprueba la existencia de un nomenclador en una determinada tabla.
Nombre:	dame_descripcion()
Descripción:	Devuelve la descripción de la tabla que se esta configurando es ese momento.

Nombre: C_Configuracion_reg	
Tipo de clase: controladora	
Atributo:	**Tipo**
Para cada responsabilidad:	
Nombre:	V_Nomencladores()
Descripción:	Permite mostrar las vistas para gestionar los nomencladores.
Nombre:	Adicionar()
Descripción:	Permite adicionar un nomenclador.
Nombre:	Eliminar()
Descripción:	Permite eliminar un nomenclador.
Nombre:	Editar()
Descripción:	Permite editar un nomenclador.
Nombre:	GestionarTabla($tabla_bd, $per_page, $controller, $view)
Descripción:	Permite mostrar un listado de nomencladores en una vista.

Nombre: M_Configuracion_Reg	
Tipo de clase: modelo	
Atributo:	**Tipo**
Para cada responsabilidad:	
Nombre:	Insertar($nomenclador,$tabla)
Descripción:	Inserta un nomenclador determinado en la tabla que le corresponde
Nombre:	Modificar($nomenclador,$tabla)
Descripción:	Modifica la descripción de un nomenclador determinado
Nombre:	Eliminar($id, $tabla)

Descripción:	Elimina un nomenclador de la tabla pasándole el id del mismo por parámetro
Nombre:	Cargar_Nomenclador($tabla)
Descripción:	Lista todos los elementos de la tabla que se le pasa por parámetros con el id como criterio de búsqueda.
Nombre:	Buscar_x_Decripcion($id, $tabla)
Descripción:	Busca y devuelve el elemento en la tabla que se pasa por parámetro con la descripción como criterio de búsqueda
Nombre:	Nomencladores_Relacionados($id_padre, $desc_id, $tabla_hijo)
Descripción:	Dado un nomenclador devuelve todos los que son dependientes de este.
Nombre:	Activar($id, $tabla)
Descripción:	Activa los nomencladores que están desactivados, es decir modifica el campo eliminado con el valor '' true''.
Nombre:	comprobar_existencia($tabla,$Descripcion)
Descripción:	Comprueba la existencia de un nomenclador en una determinada tabla.

Anexo 3. Tablas de la base de datos relacionadas con la seguridad y configuración

Nombre: empresa_importadora		
Descripción: Esta tabla es un nomenclador y almacena la descripción de cada empresa importadora		
Atributo	**Tipo**	**Descripción**
id	Integer.	Es el identificador para los datos de esta tabla.
descripcion	Varchar	Almacena la descripción de las empresas importadoras.
eliminado	tinyint	Identifica cuando una empresa importadora es eliminada.

Nombre: país_origen		
Descripción: Esta tabla es un nomenclador y almacena la descripción de cada país de origen del producto.		
Atributo	**Tipo**	**Descripción**
id	Integer.	Es el identificador para los datos de esta tabla.
descripcion	Varchar.	Almacena la descripción de los países.
eliminado	tinyint	Identifica cuando un país es eliminado.

Nombre: grupo		
Descripción: Esta tabla almacena es un nomenclador y almacena la descripción de cada grupo de alimento.		
Atributo	**Tipo**	**Descripción**
id	Integer.	Es el identificador para los datos de esta tabla.
descripcion	Varchar.	Almacena la descripción de los países.
eliminado	tinyint	Identifica cuando un país es eliminado.

Nombre: n_puerto		
Descripción: Esta tabla es un nomenclador y la misma almacena la descripción de los puertos.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
descripcion	Varchar	Es el nombre del puerto.
eliminado	Tinyint	Identifica si el puerto está o no eliminado de la aplicación.

Nombre: n_terminal		
Descripción: Esta tabla es un nomenclador y la misma almacena la descripción de las terminales.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
id_puerto	Integer	Representa la relación con la tabla n_puerto y almacena el id del puerto donde se encuentra la terminal.
descripcion	Varchar	Guarda el nombre de la terminal.
eliminado	Tinyint	Identifica si el puerto está o no eliminado de la aplicación.

Nombre: n_fumigacion		
Descripción: Esta tabla es un nomenclador y la misma almacena la descripción de productos de fumigación.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
descripcion	Varchar	Guarda el nombre del producto usado para la fumigación.

Nombre: n_certificado		
Descripción: Esta tabla es un nomenclador y la misma almacena la descripción de los certificados sanitarios		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
descripcion	Varchar	Guarda el nombre de los certificados sanitarios.
eliminado	Tinyint	Identifica si el certificado sanitario está o no eliminado de la aplicación.

Nombre: n_centro		
Descripción: Esta tabla es un nomenclador y la misma gestiona la dirección de los centros.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
localidad	Varchar	Guarda el nombre de la localidad de un municipio
id_municipio	Integer	Representa la relación con la tabla municipio y almacena el id de el municipio que esta relacionado con la dirección de un centro.
descripcion	Varchar	Almacena el nombre del centro.
calle	Varchar	Almacena el nombre o número de una calle.
numero	Integer	Representa el número que identifica al centro.
eliminado	Tinyint	Identifica si el centro está o no eliminado de la aplicación.

Nombre: n_unidad_medida		
Descripción: Esta tabla es un nomenclador y la misma gestiona las unidades de medida.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
descripcion	Varchar	Almacena el nombre de las unidades de medida.
eliminado	Tinyint	Identifica si una unidad de medida está o no eliminada de la aplicación.

Nombre: n_causa_directa		
Descripción: Esta tabla es un nomenclador y almacena las causas directas que pueden provocar una incidencia.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
descripcion	Varchar	Almacena la descripción de las causas directas de una incidencia.
eliminado	Tinyint	Identifica una causa directa como eliminada o no de la aplicación.

Nombre: n_causa_indirecta		
Descripción: Esta tabla es un nomenclador y almacena las causas indirectas que pueden provocar una incidencia.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
id_causa_directa	Integer	Representa la relación con la tabla n_causa_directa y almacena el id de la causa directa con la cual esta relacionada la causa indirecta.
descripcion	Varchar	Almacena la descripción de las causas indirectas de una incidencia.
eliminado	Tinyint	Identifica una causa indirecta como eliminada o no de la aplicación.

Nombre: n_causas_no_registro		
Descripción: Esta tabla es un nomenclador y gestiona las causas de no registro de los productos.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
descripcion	Integer	Almacena la descripción de las causas de no registro.
eliminado	Tinyint	Identifica cuando una causa de no registro es eliminada de la aplicación.

Nombre: n_conductas_seguir		
Descripción: Esta tabla es un nomenclador y gestiona las conductas a seguir ante un producto no registrado en el registro sanitario.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
descripcion	Integer	Almacena la descripción de las conductas a seguir
eliminado	Tinyint	Identifica cuando una conducta a seguir es eliminada de la aplicación.

Nombre: roles		
Descripción: Esta tabla almacena los roles que van a ser definidos en la aplicación		
Atributo	**Tipo**	**Descripción**
id_rol	Integer	Es el identificador para los datos de esta tabla.
rol	Varchar	Es el nombre de los roles definidos.

Nombre: vinculo		
Descripción: Esta tabla almacena vínculos a los cuales algunos usuarios tendrán acceso.		
Atributo	**Tipo**	**Descripción**
id_vinculo	Integer	Es el identificador para los datos de esta tabla.
desc_vinculo	Varchar	Es la descripción del vínculo.
link_ref	Varchar	Almacena el link que se les dará como acceso a los usuarios.
activo	Tinyint	Identifica si el vínculo está activo o no en la aplicación.
orden	Integer	Almacena el orden que tienen los vínculos en el menú.

Nombre: usuarios		
Descripción: Esta tabla almacena los datos de los usuarios que tienen acceso a la aplicación.		
Atributo	**Tipo**	**Descripción**
id	Integer	Es el identificador para los datos de esta tabla.
user_name	Varchar	Almacena el nombre del usuario.

password	Varchar	Almacena la contraseña que tiene el usuario.
id_rol	Integer	Representa la relación con la tabla rol y almacena el id del rol que identifica a ese usuario.
nivel	Integer	Almacena un número entero que identifica el nivel al que pertenece el usuario (nacional, provincial o municipal).
id_puerto	Integer	En caso de que el usuario sea del puerto entonces almacena el id del puerto al que pertenece.
id_aeropuerto	Integer	En caso de que el usuario sea del aeropuerto entonces almacena el id del aeropuerto al que pertenece.

GLOSARIO DE TÉRMINOS:

BSD: Licencia de software libre.

CGI: Common Gateway Interface / Interfaz Común de Pasarela.

CPHE: Centro Provincial de Higiene y Epidemiología.

CSI: Control Sanitario Estatal. Es un programa del MINSAP.

DDL: son las siglas de Data Definition Language, y se corresponde con el conjunto de órdenes que permiten definir las estructuras que van a contener los datos en un repositorio. Un subconjunto de SQL son comandos DDL. Es el formado por las órdenes CREATE, DROP y ALTER.

DML: son las siglas de Data Manipulation Language y se refiere a los comandos que permiten a un usuario manipular los datos en un repositorio, es decir, añadir, consultar, borrar o actualizar. En SQL los comandos SELECT, INSERT, UPDATE y DELETE son comandos DML.

ECMA: Ecma International es una organización internacional basada en estándares para la comunicación y la información.

Enfermedad emergente: Clase de enfermedades infecciosas que surge en lugares y momentos específicos y se convierte, o amenaza con convertirse, en una nueva epidemia.

Enfermedad exótica: Desde punto de vista sanitario se considera que una enfermedad es exótica cuando ésta no existe en el país.

ETA: Enfermedades Transmitidas por Alimentos.

Framework: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

HTML: HyperText Markup Language/Lenguaje. Lenguaje usado para escribir documentos para servidores World Wide Web.

HTTP: HyperText Transfer Protocol/ Protocolo de Transferencia de Hipertextos. Modo de comunicación para solicitar páginas Web.

Inocuidad de los alimentos: condición de los alimentos que garantiza que no causarán daño al consumidor cuando se preparen y/o consuman de acuerdo con el uso al que se destinan.

ISE: Inspección Sanitaria Estatal.

Java: Es un lenguaje de programación orientado a objetos.

JUnit: Es un conjunto de clases que permite realizar la ejecución de clases Java de manera controlada.

Linux: Es el nombre de un núcleo, pero se suele denominar con este nombre a un sistema operativo de libre distribución software libre (y de código abierto), donde el código fuente está disponible públicamente y cualquier persona, con los conocimientos informáticos adecuados, puede libremente estudiarlo, usarlo, modificarlo y redistribuirlo.

MINSAP: Ministerio de Salud Pública.

Metaprogramación: Consiste en escribir programas que escriben o manipulan otros programas (o a sí mismos) como datos, o que hacen en tiempo de compilación parte del trabajo que, de otra forma, se haría en tiempo de ejecución. Esto permite al programador ahorrar tiempo en la producción de código.

POO: Programación Orientada a Objetos. Es una técnica de programación, es la más usada actualmente en el mundo del desarrollo de software.

Salud Ocupacional: Busca proteger y mejorar la salud física, mental, social y espiritual de los trabajadores en sus puestos de trabajo.

SOAP: Siglas de Simple Object Access Protocol, es un protocolo para el intercambio de mensajes sobre redes de computadoras.

Software: Programas de sistema, utilerías o aplicaciones expresados en un lenguaje de máquina.

TCP/IP: Es un conjunto de protocolos de red en la que se basa Internet y que permiten la transmisión de datos entre redes de computadoras.

Unidad de Salud: Centro de salud que pertenece al MINSAP.

Unix: Es un sistema operativo portable, multitarea y multiusuario; desarrollado a principios de 1969 por un grupo de empleados de los laboratorios Bell.

WEB (WWW): Red de documentos HTML intercomunicados y distribuidos entre servidores del mundo entero.

XML: Sigla en inglés de Extensible Markup Language (lenguaje de marcas extensible).