UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO-CAMPUS SUR

CARRERA DE INGENIERÍA DE SISTEMAS

MENCIÓN TELEMÁTICA

ANÁLISIS, DISEÑO Y DESARROLLO DE UN MÓDULO PARA LA GESTIÓN DE MEDICAMENTOS DEL SISTEMA DE GESTIÓN MÉDICO PARA ÁREAS DE SALUD (SGMAS), PARA EL CENTRO DE SALUD NO. 3 "LA TOLA-VICENTINA" DE LA DIRECCIÓN PROVINCIAL DE SALUD DE PICHINCHA.

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SISTEMAS

CARLOS FERNANDO ARIAS MONTALVO
MARÍA JOSÉ MICHELENA PERUGACHI

DIRECTOR ING. RENÉ ARÉVALO

Quito, Octubre del 2012

DECLARACIÓN

Nosotros Carlos Fernando Arias Montalvo y María José Michelena Perugachi declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad institucional vigente.

_____ _____

Carlos Fernando Arias Montalvo María José Michelena Perugachi

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por el Sr. Arias Montalvo Carlos Fernando y la Srta. Michelena Perugachi María José, bajo mi dirección.

_____

Ing. René Arévalo

Director de tesis

AGRADECIMIENTO

Nos complace de sobre manera a través de este trabajo exteriorizar nuestro sincero agradecimiento a la Universidad Politécnica Salesiana y a sus distinguidos docentes quienes con su profesionalismo y ética puesto de manifiesto en las aulas, enrumban a cada uno de los que acudimos con sus conocimientos que nos servirán para ser útiles para la sociedad.

A Dios Todopoderoso por permitirnos terminar con éxito una etapa más en nuestra vida profesional; llenándonos de felicidad, paciencia, amor e inteligencia a lo largo de esta carrera.

A nuestro Director Ingeniero René Arévalo quien con su experiencia como docente ha sido la guía idónea durante este proceso que nos ha llevado a culminar esta tesis, brindándonos el tiempo necesario, como la información para que este anhelo llegue a ser felizmente culminado.

Carlos Fernando Arias Montalvo

María José Michelena Perugachi

DEDICATORIA

Gracias a Dios Todopoderoso por darme la vida y por la felicidad que me ha permitido brindarles a mis padres que con su esfuerzo, apoyo y confianza hicieron posible que este sueño se haga realidad.

Gracias a ustedes Carlitos y Amparito, padres míos, por creer siempre en mí, por inculcarme esos valores que llevaré siempre arraigado en mi corazón y que pondré en práctica ahora en mi vida profesional.

Gracias Sebas, Lucho, Oscarín, hermanos míos por ser mi apoyo y brindarme todo el cariño que necesite en tiempos difíciles.

Gracias a ti, Racelita, amor de mi vida, por ser mi fortaleza y ser tu quién supo darme las fuerzas necesarias para cumplir con este sueño una vez lejano y que ahora es una realidad.

Los amo a todos y esto es para ustedes…

Carlos Fernando Arias Montalvo

DEDICATORIA

Dedico el presente trabajo de Tesis en primer lugar a *Dios, por darme la oportunidad de vivir y por estar conmigo en cada paso que doy, por fortalecer mi corazón e iluminar mi mente y por haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía durante todo el periodo de estudio.*

A mis padres: Violeta Perugachi y Milton Michelena por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, gracias a sus consejos, y por el amor que siempre me han dado, además del apoyo que me brindaron para culminar mi carrera profesional. ¡Los quiero mucho!

A mi amor Alexis, por siempre estar a mi lado, brindándome todo su amor, entrega, dedicación y sobre todo tenerme mucha comprensión y paciencia durante este año de mi vida. Mil gracias porque siempre estas a mi lado sin condiciones.

Y cómo olvidar a mi compañero y amigo Carlos Arias por el tiempo dedicado a este trabajo y por todo lo que me ha enseñado en este proyecto de tesis.

Gracias a todos y cada uno de los que lean y han leído este trabajo porque, por ese simple hecho, ya forman parte de él.

María José Michelena Perugachi

PRESENTACIÓN

El desarrollo del presente proyecto tiene como finalidad el diseño y la aplicación de un módulo para la gestión de medicamentos del sistema de gestión médico para áreas de salud (SGMAS).

La creación de éste sistema, permitirá, realizar ingresos, egresos y los diferentes reportes de los medicamentos que se manejan en el centro de salud no. 3 “La Tola-Vicentina” de la Dirección Provincial de Salud de Pichincha, a través de una interfaz gráfica amigable al usuario.

La construcción del software propuesto, permitirá que la información referente a los medicamentos para su distribución gratuita sea procesada y almacenada de una forma más segura y eficiente, garantizando una mejor efectividad en el manejo del flujo y procesamiento de los datos.

Es así, como este proyecto, mediante la investigación realizada y la puesta en marcha de los conocimientos adquiridos, permitirá obtener los resultados deseados, un modelo de gestión de medicamentos, que permita mejorar la calidad de vida y la salud de la población.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS:

ÍNDICE DE TABLAS:

1. CAPÍTULO I INTRODUCCIÓN.

1.1 ANTECEDENTES.

Debido a los cambios políticos y sociales que en los últimos tiempos nuestro país ha ido experimentando y sobre todo el cambio radical que el gobierno en turno ha dado a la Salud Pública Ecuatoriana el tema de la gratuidad de los medicamentos se ha vuelto un tema de mucha controversia y debate.

Para el año 2003, el gasto total en salud fue de 1.480 millones dólares (5,5 PIB), de los cuales $524 millones corresponden a medicamentos (1,67 PIB), y de estos $403 millones era de gasto directo de los hogares.

En la Ley Orgánica de Salud se establece como obligación del Estado, garantizar a la población el acceso y disponibilidad de medicamentos de calidad a bajo costo, con énfasis en medicamentos genéricos y como aspecto más relevante el priorizar la Salud Pública sobre los intereses comerciales y económicos del país.

Así también se dispone la regulación de los gastos de publicidad y promoción a fin de que no afecten al acceso a medicamentos y a los derechos de quienes los usan. Con estos antecedentes el CONASA[1] ha planteado una serie de lineamientos generales para la Política General de Medicamentos:

- **Accesibilidad.**
- **Regulación, registro y control.**
- **Uso Racional.**

[1]CONASA – CONSEJO NACIONAL DE SALUD

Con estos lineamientos la Dirección Provincial pretende: “Diseñar un sistema de información con organismos internacionales de salud y otros países para conocer precios referenciales, proveedores de medicamentos, precios de materias primas y principios activos que se comercializan, para modernizar y transparentar el sistema de fijación de precios de medicamentos de uso humano”, cumpliendo sus funciones bajo los principios de equidad, universalidad, eficacia, calidad, calidez, solidaridad y participación social para elevar los niveles de salud y vida de la población en el marco de un ambiente natural y social saludable.

La red de servicios del Ministerio de Salud Pública (MSP) se estructura de forma regionalizada con dos niveles de descentralización: el provincial (direcciones provinciales de salud) y cantonal (áreas de salud), lamentablemente esta estructura está muy debilitada por la falta de presupuesto, a pesar de ello, se ha reconocido esta situación de crisis y están comprometidos en llevar a delante el proceso de reforma del sector a nivel central, basándose en el Decreto Ejecutivo 1014, artículo 1 que dice:

“Establecer como política pública para las Entidades de la Administración Pública Central la utilización de Software Libre en sus sistemas y equipamientos informáticos…” pudiéndose usar software propietario “…únicamente cuando no exista una solución de Software Libre que supla las necesidades requeridas, o cuando esté en riesgo la seguridad nacional, o cuando el proyecto informático se encuentre en un punto de no retorno.”, y siendo uno de los objetivos de la Dirección Provincial de Salud de Pichincha el control de la entrega oportuna de la medicación gratuita otorgada por el gobierno, es prioritario desarrollar una metodología propia para tal fin, sin olvidar que actualmente la gestión (Inventario, Administración y Control) de la medicación tiene falencias, por lo que es imperiosa la toma de decisiones acertadas y proporcionar soluciones a este problema.

Lo más importante es que la información referente a los medicamentos para su distribución gratuita sea procesada y almacenada de una forma más segura y eficiente para agilizar y garantizar el proceso de entrega, logrando un control integral de los mismos, además de proporcionar una mejor efectividad en el manejo del flujo y procesamiento de los datos.

Al establecer este sistema se verá reflejado un impacto social positivo en el Centro de Salud No. 3 “La Tola-Vicentina” de la Dirección Provincial de Salud de Pichincha, el cual proporcionará una información confiable, agilizando y facilitando el trabajo en el proceso de Gestión de Medicamentos, a fin de apoyar el proceso de generar un modelo de gestión, atención y financiamiento que sea sustentable y sostenible, orientado a satisfacer a la demanda con criterio de equidad, universalidad, calidad, calidez, eficacia y efectividad con personal altamente competente para mejorar la calidad de vida y la salud de la población.

1.2 DESCRIPCIÓN DEL PROBLEMA.

La Dirección Provincial de Salud al ser una entidad Pública de impacto social necesita de procesos sistemáticos para llevar un control adecuado de todos los recursos que posee siendo una de sus prioridades la entrega de medicamentos gratuitos para la población.

Lamentablemente la información que se posee actualmente se trabaja de forma manual, ya que no cuenta con las herramientas necesarias para llevar un control adecuado de los medicamentos que llegan al Centro de Salud No. 3 “La Tola-Vicentina” de la Dirección Provincial de Salud de Pichincha; esto conlleva a un deficiente uso de inventario, control y administración de los mismos.

Actualmente se lleva el inventario de los medicamentos únicamente en formato plano y no es estandarizado, el mismo que presenta varias falencias ya que no

tienen un control apropiado y seguro de los medicamentos existentes, tanto de los ingresados como de los despachados, dificultando de esta manera al personal encargado el llevar un registro de cuales medicamentos hay en stock, cuales hacen falta adquirirlos, cuales están por terminarse, e incluso cuales están por caducarse y deben ser expendidos con mayor rapidez o dado el caso devueltos a los respectivos proveedores para su reposición.

Además, se pretende eliminar manos inescrupulosas que hagan mal uso de los medicamentos al mantener un adecuado y correcto inventario de los mismos.

Al analizar las necesidades de automatización que tiene el Centro de Salud No. 3 “La Tola-Vicentina” y aprovechando la infraestructura que posee la Dirección Provincial de Salud de Pichincha en cuantos a equipos de telecomunicaciones, con enlaces propios que conectan con las cabeceras de Áreas de Salud de Pichincha, se pretende automatizar ciertos procesos que permitan mejorar el trabajo de las distintas áreas al desarrollar un SISTEMA DE GESTIÓN MÉDICO PARA LAS ÁREAS DE SALUD PÚBLICA DE PICHINCHA (SGMAS), formado por los siguientes módulos :

- Módulos de turnos y cita previa
- Módulo de usuario e historia clínica (adulto y adulto mayor)
- Módulo de historia clínica de menores de 5 años
- Módulo de gestión medicamentos
- Módulo de farmacia
- Módulo de recursos humanos
- Módulo de laboratorio
- Módulo de vacunas

Dichos módulos fueron analizados debido a la necesidad de automatizar todos los procesos que se realizan en cada Área de Salud para mejorar la atención médica y el trabajo administrativo del personal de RRHH.

Para el desarrollo de este proyecto se tendrá como área piloto al Centro de Salud No. 3 “La Tola-Vicentina”, de donde se recopilará la información necesaria para tener un estándar a nivel de todas las áreas del Ministerio de Salud Pública, ya que cuenta con el apoyo de la Dirección Provincial de Salud de Pichincha.

El objetivo que se persigue con la implementación del sistema es:

Agilizar el proceso de gestión de medicamentos:

En donde se acelere el proceso de entrega de medicamentos, la administración y control de los mismos.

Cobertura global y portabilidad:

El diseño de la aplicación estará orientado a la plataforma de Internet, lo que le permite tener presencia global; es decir, se podrá acceder desde cualquier punto de la Provincia de Pichincha en cada Centro y Subcentro de Salud que este regido por la Dirección Provincial de Salud de Pichincha, con sólo disponer de una conexión a Internet o simplemente con la utilización de la Intranet.

Registro de operaciones:

Se registrará cada una de las operaciones que se realizan en el sistema identificando al usuario, fecha, hora y el tipo de transacción llevada a cabo; de esta manera, se podrá mantener un control detallado y una auditoría sobre uso del sistema.

Uso de plataforma de Software Libre:

Al impulsar el uso de herramientas gratuitas y software libre por parte del Estado Ecuatoriano las herramientas a utilizarse será PHP como BackEnd y como gestor de Base de Datos Postgres siendo estos de distribución libre.

1.3 RECOPILACIÓN DE INFORMACIÓN PARA LA IMPLEMENTACIÓN DEL SISTEMA.

La Dirección Provincial de Salud de Pichincha, es una entidad de gestión del Ministerio de Salud Pública.

Las Áreas de Salud son dependencias públicas que forman parte de la Dirección Provincial de Salud de Pichincha, que actúan como organizaciones que prestan servicios enfocados a dar atención médica gratuita a la población del sector. Las áreas de salud tiene entre sus funciones: planificar y ordenar las actividades que se desarrollan en los centros de salud, con el fin de mejorar la calidad de atención a los pobladores del sector de acuerdo a una mejor distribución de los servicios y beneficios que prestan las unidades médicas mencionadas anteriormente, por esta razón, la Dirección Provincial de Salud, coordina su trabajo con los centros de salud y brigadas para articular y sistematizar las actividades de las distintas instancias zonales y sus relaciones con los sectores públicos y privados.

1.3.1 VISIÓN.

Somos una institución que superando la vulnerabilidad institucional ha pasado a la sostenibilidad institucional, que promueve con sus políticas una provincia saludable, que ejerce rectoría y asesoramiento en el ámbito de la coordinación, organización, conducción y regulación del sistema provincial de salud con enfoque plural, intercultural y de género en el ámbito de la

promoción y la protección de la salud, con ambientes y estilos de vida saludables, generar un modelo de gestión, atención y financiamiento que es sustentable y sostenible, orientado a satisfacer a la demanda con criterio de equidad, universalidad, calidad, calidez, eficacia y efectividad con personal altamente competente bien remunerado y que trabaja con tecnología de punta para mejorar la calidad de vida y la salud de la población.

1.3.2 MISIÓN.

En Pichincha somos la Institución pública rectora, reguladora, proveedora y coordinadora de la salud de la provincia. Brindamos servicio de salud a través de nuestra red, que está constituida por los hospitales y áreas de salud que coordinan acciones con otras instituciones públicas, semipúblicas y privadas, para promoción, prevención, curación y rehabilitación de la salud, con enfoque integral, cultural y de género; bajo los principios de equidad, universalidad, eficacia, calidad, calidez, solidaridad y participación social para elevar los niveles de salud y vida de la población en el marco de un ambiente natural y saludable.

1.3.3 ORGANIGRAMA ESTRUCTURAL DEL MINISTERIO DE SALUD PÚBLICA.

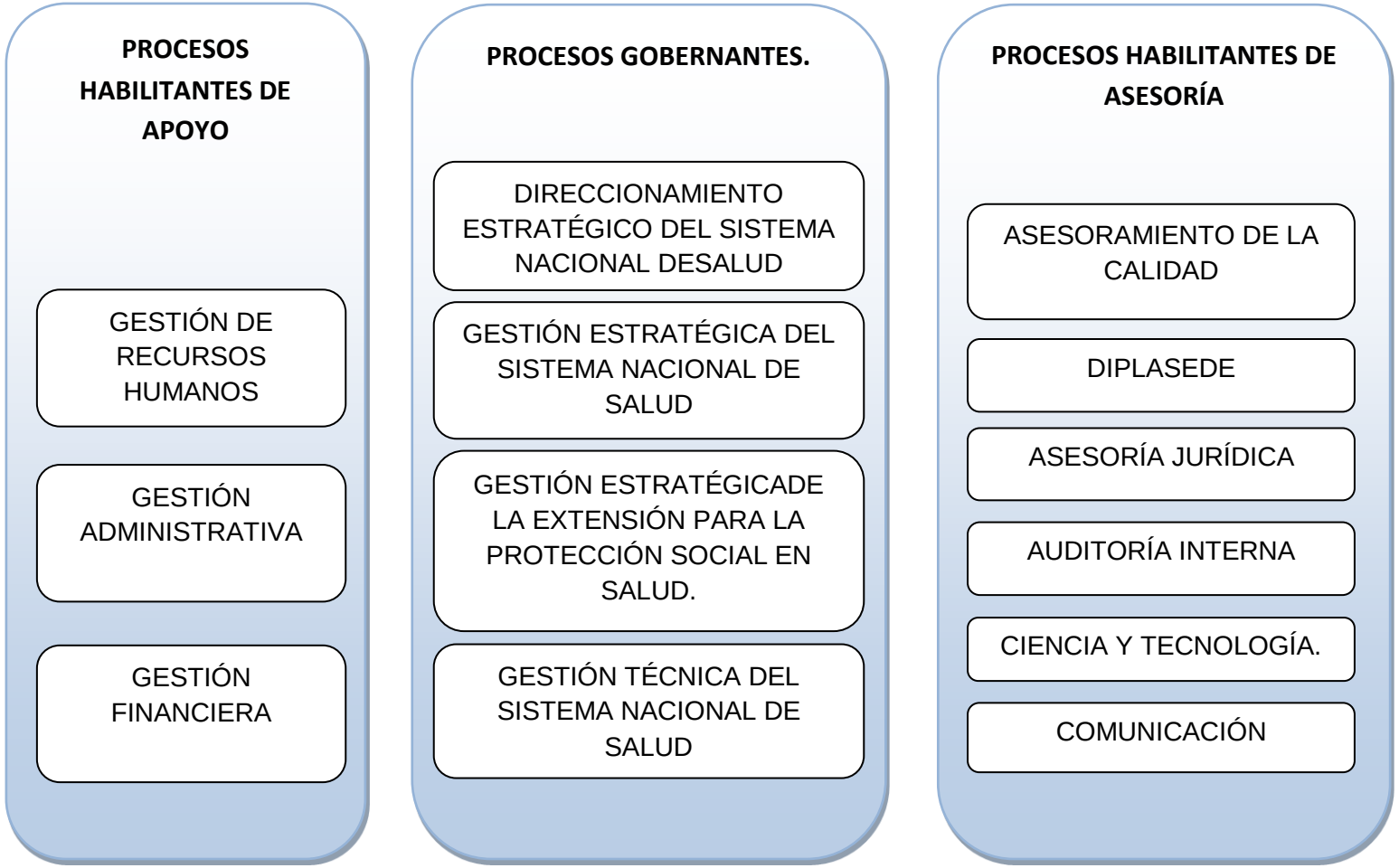


Figura1.1: Organigrama Estructural Ministerio de Salud Pública[2]

[2] *Figura1.1: Organigrama Estructural Ministerio de Salud Pública*

1.3.4 PROCESOS PRINCIPALES DE LA DIRECCIÓN PROVINCIAL DE SALUD.

La Dirección Provincial de Pichincha se maneja por los siguientes procesos textuales:

- Definir las políticas, estrategias y lineamientos para las operaciones de los centros de salud que se encuentran dentro de su justificación.
- Mantener la vigilancia en la calidad de la salud pública que se presenta a la comunidad a través de los distintos programas que se ejecutan en la organización.
- Desarrollar planes para promocionar la salud en sectores y población de acceso limitado controlando la programación de enfermedades controlables.
- Provisión de servicios.
- Mejorar e implementar estrategias que permitan un mejor y eficiente control sanitario para evitar posibles incidentes en los centros de salud.
- Incentivar a los profesionales para el progreso en el área de salud a través del desarrollo de investigaciones de carácter científico y tecnológico.

1.4 METODOLOGÍA.

La metodología XP (Extreme Programing), ó programación extrema es una metodología ligera y ágil para el desarrollo de software eficiente y altamente efectivo, se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado.

Uno de los principales objetivos de la metodología XP es la satisfacción del cliente en cuanto le proporciona el software que necesita y cuando lo necesita, respondiendo muy rápido a las necesidades del cliente.

1.4.1 ROLES EN LA METODOLOGÍA XP.

1.4.1.1 Programador.

- Más responsabilidad que en otros modos de desarrollo.
- Responsable sobre el código.
- Responsable sobre el diseño (refactorización, simplicidad).
- Responsable sobre la integridad del sistema (pruebas).
- Capacidad de comunicación (pair-programming).
- Acepta críticas (código colectivo).

1.4.1.2 Cliente.

- Define especificaciones (Historias de Usuario).
- Influye sin controlar.
- Confía en el grupo de desarrollo.
- Define pruebas funcionales.

1.4.1.3 Encargado de Pruebas.

- Apoya al cliente en la preparación/realización de las pruebas funcionales.
- Ejecuta las pruebas funcionales y publica los resultados.

1.4.1.4 Encargado de Seguimiento (Tracker).

- Recoge, analiza y publica información sobre la marcha del proyecto sin afectar demasiado el proceso.
- Supervisa el cumplimiento de las estimaciones en cada iteración.
- Informa sobre la marcha de la iteración en curso.
- Controla la marcha de las pruebas funcionales, de los errores reportados, de las responsabilidades aceptadas y de las pruebas añadidas por los errores encontrados.

1.4.1.5 Entrenador (Coach).

- Responsable del proceso y de proveer guías necesarias a los miembros del equipo, de forma que se aplique correctamente la metodología XP.
- Identifica las desviaciones y reclama atención sobre las mismas.
- Interviene directamente si es necesario.
- Solucionar rápidamente el problema.

1.4.1.6 Consultor.

- Apoya al equipo XP en cuestiones puntuales.

1.4.1.7 Jefe del Proyecto.

- Favorece la relación entre clientes y desarrolladores.
- Cubre las necesidades del equipo.
- Coordina y asegura que se alcancenlos objetivos.

1.4.2 FASES DE LA METODOLOGÍA XP.

El ciclo de desarrollo de la Metodología XP consiste en los siguientes pasos:

1. El cliente define el valor del proyecto a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye el proyecto.
5. Vuelve al paso 1.

En todas estas iteraciones tanto el cliente como el programador están en constante contacto, no obstante, no se debe presionar al programador a realizar más trabajo que el establecido, ya que esto podría ocasionar pérdida en la calidad del software y no se cumplirán los plazos.

El ciclo de vida ideal de XP consiste en las siguientes fases:

1.4.2.1 Fase I: Exploración.

En esta primera fase los clientes plantean las principales necesidades que se desea cubrir con el desarrollo del programa (historias de usuario), las cuales son importantes para la primera entrega del producto.

Además, el equipo de programadores se familiariza con las herramientas que se utilizarán en el proyecto, se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.

Esta fase puede tomar semanas a pocos meses dependiendo del tamaño, complejidad y de la familiaridad que tengan los programadores con la tecnología.

1.4.2.2 Fase II: Planificación de la Entrega.

Se establece la prioridad de cada historia de usuario, se realiza una estimación del esfuerzo necesario para la implementación de cada una de ellas.

Se estima el riesgo y cuánto tiempo conllevará la implementación, se fija el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.

En cuanto el software aporte algo al proyecto debemos de tener lista las primeras versiones y crear unidades de prueba de cada módulo que se desarrolle para que esto funcione correctamente.

1.4.2.3 Fase III: Iteraciones.

Se realizan varias iteraciones sobre el sistema antes de ser entregado, las cuales no deben durar más de tres semanas. En la primera iteración se establece una arquitectura del sistema que va hacer utilizada durante el resto del proyecto, mediante las historias de usuarios que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración, al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación y tareas no superadas en la iteración anterior.

Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas tiene un responsable, pero se las lleva a cabo por parejas de programadores.

1.4.2.4 Fase IV: Producción.

En esta fase de producción se requiere realizar pruebas adicionales y revisiones de rendimiento del sistema antes de que sea trasladado al entorno del cliente. Además, se puede realizar la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Las ideas que han sido propuestas y las sugerencias tienen que ser documentadas para su posterior implementación.

1.4.2.5 Fase V: Mantenimiento.

En esta fase de mantenimiento se puede realizar cambios en la estructura, se puede requerir nuevo personal dentro del equipo.

Es importante mientras la primera versión del proyecto XP se encuentra en producción, mantener el sistema en funcionamiento al

mismo tiempo que se desarrolla nuevas iteraciones, de esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción.

1.4.2.6 Fase VI: Muerte del Proyecto.

Se llega a esta fase cuando el cliente no tiene más historias para ser incluidas en el sistema, se genera la documentación final del sistema y no se realizan más cambios en la arquitectura.
También puede ocurrir cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

1.4.3 PRÁCTICAS XP.

Mediante las prácticas que se describen a continuación se disminuye la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione.

1.4.3.1 Prácticas de codificación. [a]

- Simplicidad de código y de diseño para producir software fácil de modificar.
- Reingeniería continúa para lograr que el código tenga un diseño óptimo.
- Desarrollar estándares de codificación, para comunicar ideas con claridad a través del código.
- Desarrollar un vocabulario común, para comunicar las ideas sobre el código con claridad.

---

[a] www.navegopolis.net/files/artículos/gestion_y_procesos.pdf

1.4.3.2 Prácticas de desarrollo.

- Adoptar un método de desarrollo basado en las pruebas para asegurar que el código se comporta según lo esperado.
- Programación por parejas, para incrementar el conocimiento, la experiencia y las ideas.
- Asumir la propiedad colectiva del código, para que todo el equipo sea responsable de él.
- Integración continua, para reducir el impacto de la incorporación de nuevas funcionalidades.

1.4.3.3 Prácticas de negocio.

- Integración de un representante del cliente en el equipo, para encauzar las cuestiones de negocio del sistema de forma directa, sin retrasos o pérdidas por intermediación.
- Adoptar una planificación para centrar en la agenda el trabajo más importante.
- Entregas regulares y frecuentes para satisfacer la inversión del cliente.
- Ritmo de trabajo sostenible, para terminar la jornada cansado pero no agotado.

1.4.4 VENTAJAS METODOLOGÍA XP:

- Ahorro de mucho tiempo y muchos recursos.
- Se consiguen productos usables con mayor rapidez.
- El proceso de integración es continuo, por lo que el esfuerzo final para la integración es nulo. Se consigue integrar todo el trabajo con mucha mayor facilidad.
- Se atienden las necesidades del usuario con mayor exactitud. Esto se consigue gracias a las continuas versiones que se ofrecen al usuario.

- Se consiguen productos más fiables y robustos contra los fallos gracias al diseño de los test de forma previa a la codificación.
- Obtenemos código más simple y más fácil de entender, reduciendo el número de errores.

1.4.5 DESVENTAJAS METODOLOGÍA XP:

- Es recomendable emplearlo solo en proyectos a corto plazo.
- Existe menos generación de documentación de soporte del sistema.

1.4.6 BENEFICIOS METODOLOGÍA XP.

- El cliente tiene el control sobre las prioridades.
- Se hacen pruebas continuas durante el proyecto.
- Los clientes son claves en los procesos de desarrollo.
- Los procesos se aceptan y se acuerdan, no se imponen.
- Desarrolladores y gerentes comparten el liderazgo del proyecto.
- El trabajo de los desarrolladores con las personas que conocen el negocio es regular, no puntual.

2. CAPÍTULO II ANÁLISIS Y REQUERIMIENTOS

2.1 ESPECIFICACIONES DE REQUERIMIENTOS INÍCIALES.

La Dirección Provincial de Salud de Pichincha es la institución pública rectora, reguladora, proveedora y coordinadora de la salud de la provincia. Uno de los principales elementos que conforman su misión es la de brindar servicios de salud a través de su red, que está constituida por hospitales y áreas de salud coordinando acciones con otras instituciones públicas, semipúblicas y privadas, para la promoción, prevención, curación y rehabilitación de la salud, con enfoque, integral, cultural y de género.

La Dirección Provincial cumple sus funciones bajo los principios de equidad, universalidad, eficacia, calidad, calidez, solidaridad y participación social para elevar los niveles de salud y vida de la población en el marco de un ambiente natural y social saludable[3].

Figura 2.1: Dirección Provincial de Pichincha[3]

[3] *Fuente y Figura 2.1: Dirección Provincial de Pichincha*

Siendo una institución que superando la vulnerabilidad institucional ha pasado a la sostenibilidad institucional, que promueve con sus políticas una provincia saludable, que ejerce rectoría y asesoramiento en el ámbito de la coordinación, organización, conducción y regulación del sistema provincial de salud con enfoque plural, intercultural y de género en el ámbito de la promoción y la protección de la salud, con ambientes de vida saludables, generar un modelo de gestión, atención y financiamiento que sea sustentable y sostenible, orientado a satisfacer a la demanda con criterio de equidad, universalidad, calidad, calidez, eficacia y efectividad con personal altamente competente bien remunerado y mejorar la calidad de vida y la salud de la población.

Lamentablemente la información que posee actualmente se la procesa de forma manual, por lo que se ha buscado la manera de implementar un proceso de sistematización, aprovechando la infraestructura que posee la Dirección Provincial de Salud de Pichincha en materia de comunicación, con enlaces propios que conectan con las cabeceras de Áreas de Salud de Pichincha, se pretende automatizar ciertos procesos que permitan mejorar el trabajo de las distintas áreas al desarrollar un SISTEMA DE GESTIÓN MÉDICO PARA LAS ÁREAS DE SALUD PÚBLICA DE PICHINCHA (SGMAS), siendo uno de sus módulos el perteneciente a este proyecto de tesis que es el de **GESTIÓN DE MEDICAMENTOS**.

Dicho módulo fue analizado debido a la necesidad de automatizar los procesos de ingreso, bodegaje y despacho de la medicación que ingresa a los diferentes establecimientos de salud pública, para que permitan mejorar la atención médica a nivel de áreas de salud y el trabajo administrativo.

Siendo área piloto el Centro de Salud No. 3 La Tola – Vicentina de la Dirección Provincial de Salud de Pichincha, donde se recopiló la información necesaria para la realización del módulo y tomando en cuenta que es uno de los pocos establecimientos de salud bajo la Dirección Provincial de Salud de Pichincha que posee un proceso de gestión

informática ordenada y liderado por una sola persona que cumple el perfil (Ingeniero en Sistemas), todos los análisis para el desarrollo del proyecto se los ejecutó en el nombrado establecimiento.

Basándose en el Decreto Ejecutivo 1014, artículo 1 que dice:

"*Establecer como política pública para las Entidades de la Administración Pública Central la utilización de Software Libre en sus sistemas y equipamientos informáticos…" pudiéndose usar software propietario "…únicamente cuando no exista una solución de Software Libre que supla las necesidades requeridas, o cuando esté en riesgo la seguridad nacional, o cuando el proyecto informático se encuentre en un punto de no retorno."*, y siendo uno de los objetivos de la Dirección Provincial de Salud de Pichincha el control de la entrega oportuna de la medicación gratuita otorgada por el gobierno, es prioritario desarrollar una metodología propia para tal fin, sin olvidar que actualmente la gestión (Inventario, Administración y Control) de la medicación tiene falencias, por lo que es imperiosa la toma de decisiones acertadas y proporcionar soluciones a este problema.

Lo más importante es que la información referente a los medicamentos para su distribución gratuita sea procesada y almacenada de una forma más segura y eficiente para agilizar y garantizar el proceso de entrega, logrando un control integral de los mismos, además de proporcionar una mejor efectividad en el manejo del flujo y procesamiento de los datos.

Al establecer este sistema se verá reflejado un impacto social positivo en el Centro de Salud No. 3 "La Tola-Vicentina" de la Dirección Provincial de Salud de Pichincha, el cual proporcionará una información confiable, agilizando y facilitando el trabajo en el proceso de Gestión de Medicamentos.

2.2 ANÁLISIS DE REQUERIMIENTOS.

En todo desarrollo de sistemas de software es de suma importancia el seguir alguna especificación que permita a los desarrolladores disponer de una metodología de desarrollo a seguir que incluya todas las etapas del desarrollo del sistema, desde la pesquisa inicial de requerimientos hasta las pruebas finales del sistema.

La herramienta de software que este proyecto propone desarrollar es una aplicación que permita llevar un correcto control de los procesos de gestión de medicamentos en el Centro de Salud No. 3 La Tola – Vicentina de la Dirección Provincial de Salud de Pichincha, para garantizar que se los realice de manera transparente y eficaz.

El capítulo en sí proporciona una pequeña introducción a lo que es la disciplina de la ingeniería de software, y posteriormente detalla los procesos y principios de análisis y de diseño del software que son utilizadas para complementar el desarrollo del sistema que se propone.

2.2.1 INGENIERÍA DE SOFTWARE

Ingeniería de Software es una disciplina o área de las ciencias de la computación que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelve problemas de todo tipo, trata con áreas muy diversas de las ciencias de la computación, tales como construcción de compiladores, sistemas operativos, o desarrollos en Internet como es el caso de la aplicación de software de esta propuesta. La Ingeniería de Software abarca todas las fases del ciclo de vida del desarrollo de cualquier tipo de sistema de información aplicable a áreas tales como los negocios, investigación científica, medicina, producción, logística, banca, etc.

Un aspecto muy importante de Ingeniería de Software es que proporciona parámetros formales para lo que se conoce como Gestión (o Administración) de Proyectos de Software, que proporcionan diversas métricas y metodologías que pueden usarse como especificaciones para todo lo referente a la administración del personal involucrado en proyectos de software, ciclos de vida de un proyecto, costos, y en sí todo el aspecto administrativo que implica el desarrollar software.

De acuerdo con Pressman[4], Ingeniería en general es el análisis, diseño, construcción, verificación y gestión de entidades técnicas. En general, todo proceso de ingeniería debe comenzar por contestar las siguientes preguntas: ¿Cuál es el problema a resolver?, ¿Cuáles son las características de la entidad que se utiliza para resolver el problema?, ¿Cómo se realizará la entidad (y la solución)?, ¿Cómo se construirá la entidad?, ¿Cómo va a probarse la entidad?, y ¿Cómo se apoyará la entidad cuando los usuarios finales soliciten correcciones y adaptaciones a la entidad?
Para los fines que se desarrolla el software propuesto dentro de este proyecto, podemos contestar estas preguntas en primera instancia desde un punto de vista global y sin considerar detalles específicos, con los siguientes argumentos:

Desarrollar una aplicación de software que permita llevar un correcto control de los procesos de gestión de medicamentos para garantizar que se los realice de manera transparente y eficaz.

La plataforma para el desarrollo del software es la siguiente:

- PHP como lenguaje de programación.
- Postgresql como Motor de Base de Datos.

[4]Ingeniería de Software Un Enfoque Práctico (**Pressman** 5th Ed)

La aplicación de software deberá ser probada en un intervalo de tiempo adecuado y lo suficientemente amplio como para poder obtener retroalimentación por parte de los usuarios y hacer las correcciones pertinentes.

El software deberá estar documentado adecuadamente para facilitar futuros procesos tales como posibles expansiones ó adaptaciones a nuevas exigencias por parte de los usuarios finales.

El modelo de ingeniería de software que esta tesis sigue es el modelo espiral, que será descrito a continuación.

2.2.2 Modelo Espiral

El modelo en espiral es una de las metodologías más recomendables para el desarrollo y creación de un programa, ya que consta de pocas etapas o fases, las cuales se van realizando de una manera continua y cíclica.

Figura 2.2 Modelo Espiral[5]

[5] *Figura 2.2 Modelo Lineal Secuencial: http://es.scribd.com/doc/11468208/Modelo-Espiral*

Para el desarrollo del módulo de gestión de medicamentos se tomó la decisión de trabajar con este modelo por ajustarse más a las necesidades del sistema y guarda relación con la aplicación de la XP; posee las siguientes ventajas:

- Puede adaptarse y aplicarse a cada etapa de ciclo de vida del software.
- Como el software evoluciona a medida que progresa el proceso de desarrollo, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos.
- El modelo en espiral permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto.
- El modelo en espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en problemas.

Cada ciclo de la espiral se divide en 4 etapas:

2.2.2.1 Definición de objetivos:

Para esta fase del proyecto se definen los objetivos específicos. Se identifican las restricciones de los procesos y el producto, y se estipula un plan detallado de administración. Se identifican los riesgos del proyecto. Dependiendo de esos riesgos, se planean estrategias alternativas.

2.2.2.2 Evaluación y reducción de riesgos:

Se lleva a cabo un análisis detallado para cada uno de los riesgos del proyecto. Se definen los pasos para reducir dichos riesgos.

2.2.2.3 Desarrollo y validación:

Teniendo como base la evaluación de riesgos, se elige un modelo apropiado para el desarrollo del sistema, por ejemplo, si mediante la evaluación de los riesgos determinamos que los riesgos de protección es la principal consideración, un modelo de desarrollo apropiado, seria basado en implementación de seguridades de acceso a la información, y así sucesivamente.

2.2.2.4 Planeación:

El proyecto se revisa y se toma la decisión, de si se debe continuar con un ciclo posterior de la espiral. Si se decide continuar, se debe planear la siguiente fase del proyecto.

2.2.3 HISTORIAS DE USUARIOS

El primer paso a seguir con la metodología XP es elaborar historias de usuario, las cuales nos permite describir los requisitos del sistema. Son formatos muy pequeños, usadas para estimar tiempos de desarrollo del sistema, también se verifica en la fase de pruebas para afirmar que el programa cumple con la historia de usuario específica.

	HISTORIA DE USUARIO
Usuario:	**Número:**
Nombre Historia:	**Riesgo en Desarrollo (Alta/Media/Baja):**
Referencia a la Historia de usuario número:	**Prioridad en Negocio (Alta/Media/Baja):**
Descripción:	
Observaciones:	

Tabla 2.2.3: Plantilla Historia de Usuario[6]

[6] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

2.2.3.1 Historia de Usuario: Requerimientos Iníciales Software

HISTORIA DE USUARIO	
Usuario: Ing. Richard Murillo, Ing. Maritza Badillo	**Número:** 1
Nombre Historia: Requerimientos Iníciales Software.	**Riesgo en Desarrollo (Alta/Media/Baja):** Media
Referencia a la Historia de usuario número: 1	**Prioridad en Negocio (Alta/Media/Baja):** Alta
Descripción:	**Requerimientos Iníciales:** Desarrollar un software que permita el Análisis, diseño y desarrollo del módulo de Gestión de Medicamentos para el "SISTEMA DE GESTIÓN MÉDICO PARA ÁREAS DE SALUD" (SGMAS), para el Centro de Salud No. 3 "La Tola-Vicentina" de la Dirección Provincial de Salud de Pichincha. Desarrollar una hoja de estilos para el "SISTEMA DE GESTIÓN MÉDICO PARA ÁREAS DE SALUD", de fácil navegación y utilización para el usuario final.
Observaciones:	Presentar un diseño de navegabilidad. Indicar con que otros módulos del sistema interactuaría el módulo Gestión de Medicamentos.

Tabla2.2.3.1 Historia de Usuario: Requerimientos Iníciales Software[7]

[7] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

2.2.3.2 Historia de Usuario: Requerimientos Iníciales Gestión Medicamentos

<table>
<tr><th colspan="3">HISTORIA DE USUARIO</th></tr>
<tr><td colspan="2">Usuario:Ing. Richard Murillo</td><td>Número:2</td></tr>
<tr><td colspan="2">Nombre Historia: Requerimientos Iníciales Gestión Medicamentos.</td><td>Riesgo en Desarrollo (Alta/Media/Baja): Media</td></tr>
<tr><td colspan="2">Referencia a la Historia de usuario número: 1</td><td>Prioridad en Negocio (Alta/Media/Baja): Alta</td></tr>
<tr><td>Descripción:</td><td colspan="2">Requerimientos Iníciales:
Este software debe automatizar los procesos internos que se realizan en bodega:

• Ingresar los datos y características de los medicamentos.
• Realizar despachos de medicamentos a farmacia, lo cual permita llevar un control eficiente y seguro de los mismos.
• Ingresar los medicamentos por lotes de medicación, para ser despachados conforme van ingresando al sistema.
• Generar reportes, que ayuden a manejar el inventario y control de medicamentos existentes.</td></tr>
<tr><td>Observaciones:</td><td colspan="2">Revisar software con el que están trabajando actualmente para realizar este proceso en Bodega.</td></tr>
</table>

Tabla2.2.3.2: Historia de Usuario: Requerimientos Iníciales Gestión Medicamentos[8]

[8] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

2.2.3.3 Historia de Usuario: Diseño Base de datos

	HISTORIA DE USUARIO
Usuario: Ing. Richard Murillo	**Número:**3
Nombre Historia: Diseño Base de datos	**Riesgo en Desarrollo (Alta/Media/Baja):** Alta
Referencia a la Historia de usuario número: 2	**Prioridad en Negocio (Alta/Media/Baja):** Alta
Descripción:	Diseñar el modelo conceptual de la base de datos, que permita obtener acceso a información exacta y actualizada. Que no exista duplicidad de información. Permita llevar un control real de la medicación existente, a los lugares que fue destinada, registro de bajas de medicación. Llevar un registro de las características de los medicamentos mediante el manejo de un código CUM (*CÓDIGO ÚNICO DE MEDICAMENTOS*).
Observaciones:	Diseñar el diagrama de la base de datos.

Tabla2.2.3.3.: Historia de Usuario: Diseño Base de datos[9]

[9] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

2.2.3.4 Historia de Usuario: Corrección base de datos

	HISTORIA DE USUARIO
Usuario: Ing. Richard Murillo	**Número:** 4
Nombre Historia: Corrección base de datos	**Riesgo en Desarrollo (Alta/Media/Baja):** Baja
Referencia a la Historia de usuario número: 3	**Prioridad en Negocio (Alta/Media/Baja):** Alta
Descripción:	Rediseñar el modelo conceptual de la base de datos Se debe crear tablas de mantenimiento para parametrizar el sistema. Generar el modelo físico de la base para observar con mayor claridad las relaciones y generar el script para observar que no existan errores.
Observaciones:	Modificar algunas tablas, para el manejo adecuado de la información.

Tabla2.2.3.4.: Historia de Usuario: Corrección Base de datos[10]

[10] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

2.2.3.5 Historia de Usuario: Presentación Prototipo

HISTORIA DE USUARIO	
Usuario: Ing. Richard Murillo	**Número:** 5
Nombre Historia: Presentación Prototipo	**Riesgo en Desarrollo (Alta/Media/Baja):** Baja
Referencia a la Historia de usuario número: 2	**Prioridad en Negocio (Alta/Media/Baja):** Alta
Descripción:	Presentación de un bosquejo de la hoja de estilos, para estandarizar colores, botones, interfaz. Presentación del menú del sistema.
Observaciones:	La interfaz de usuario debe ser fácil de manejar.

Tabla2.2.3.5.: Historia de Usuario: Presentación Prototipo[11]

[11] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

2.2.3.6 Historia de Usuario: Desarrollo Generar Orden de Despacho

	HISTORIA DE USUARIO
Usuario: Ing. Richard Murillo	**Número:** 6
Nombre Historia: Desarrollo Generar Orden de Despacho	**Riesgo en Desarrollo (Alta/Media/Baja):** Media
Referencia a la Historia de usuario número: 2	**Prioridad en Negocio (Alta/Media/Baja):** Alta
Descripción:	Para generar una orden de despacho de medicamentos, se debe llevar un registro de datos como: (Nombre de la bodega, Transacción, Destinatario, Fecha Movimiento, Estado Movimiento, Núm. Referencia), para poder llevar un control adecuado del lugar que se despacha la medicación. El sistema debe permitir al responsable de bodega, escoger de una lista de medicamentos cuales van hacer despachados, la cantidad y automáticamente actualizar el stock. Es importante adicionar el tipo de programa al cual pertenece la medicación. El despacho de medicamentos se realizará conforme van ingresando al sistema. Debe generar una orden de despacho de los medicamentos, y poder imprimirla.
Observaciones:	Analizar la manera de capturar los datos y generar una orden de despacho para poder imprimirla.

Tabla2.2.3.6.: Historia de Usuario: Generar Orden de Despacho.[12]

[12] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

2.2.3.7 Historia de Usuario: Desarrollo Ingreso de Medicamentos

	HISTORIA DE USUARIO
Usuario: Ing. Richard Murillo	**Número:** 7
Nombre Historia: Desarrollo Ingreso de Medicamentos	**Riesgo en Desarrollo (Alta/Media/Baja):** Media
Referencia a la Historia de usuario número: 2	**Prioridad en Negocio (Alta/Media/Baja):** Alta
Descripción:	El sistema debe ingresar los datos de cada medicamento (unidad de medida, concentración, familia, presentación comercia, etc.), Para poder identificar un medicamento se realizará mediante su nombre o mediante un CUM, (código único de medicamento). Se debe registrar stock máximo y mínimo de los medicamentos. Llevar un control de registro de bajas, y el motivo por el cual fue ocasionado.
Observaciones:	Familiarizarse con los diferentes términos para el registro de un medicamento (Unidad de medida, concentración, familia, etc.). Presentar interfaz.

Tabla2.2.3.7.: Historia de Usuario: Desarrollo Ingreso de Medicamentos.[13]

[13] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

2.2.3.8 Historia de Usuario: Inicio de sesión usuarios de Bodega.

	HISTORIA DE USUARIO
Usuario: Ing.Richard Murillo	**Número:** 8
Nombre Historia: Inicio de sesión usuarios de Bodega.	**Riesgo en Desarrollo (Alta/Media/Baja):** Baja
Referencia a la Historia de usuario número: 2	**Prioridad en Negocio (Alta/Media/Baja):** Media
Descripción:	El software debe contar con una interfaz que permita ingresar el usuario y contraseña y poder acceder al sistema.
Observaciones:	Se deberá tener en cuenta que existe un módulo específico para la creación de usuarios dentro del sistema "SISTEMA DE GESTIÓN MÉDICO PARA ÁREAS DE SALUD" (SGMAS), pero dentro del Módulo Gestión de Medicamentos se realizará la interfaz de usuarios para tener un control de ingreso al sistema.

Tabla2.2.3.8.: Historia de Usuario: Inicio de sesión usuarios de Bodega.[14]

[14] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

2.2.3.9 Historia de Usuario: Desarrollo Ingreso Lote de Medicación.

	HISTORIA DE USUARIO
Usuario: Ing Richard Murillo	**Número:** 9
Nombre Historia: Desarrollo Ingreso Lote de Medicación.	**Riesgo en Desarrollo (Alta/Media/Baja):** Media
Referencia a la Historia de usuario número: 2	**Prioridad en Negocio (Alta/Media/Baja):** Alta
Descripción:	Los medicamentos deben ser ingresados por número de lotes, lo cual permite una mejor distribución de los mismos. Además se debe llevar un registro de la fecha de ingreso de la medicación, la fecha de elaboración de los medicamentos, con su respectiva fecha de caducidad. Llevar un control de costo de Lote y la cantidad ingresada por lote. El stock debe de actualizarse automáticamente después de cada ingreso de medicación. Es importante registrar el ingreso de lote de medicación con un número de documento contable, para llevar un correcto control de quien entrega la medicación, la factura de la cual procede.
Observaciones:	Presentar interfaz con los avances realizados.

Tabla 2.2.3.9. Historias de Usuario: Desarrollo Lote de Medicación.[15]

[15] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

2.2.3.10 Historia de Usuario: Generación de Reportes

		HISTORIA DE USUARIO
Usuario: Ing. Richard Murillo		**Número:**10
Nombre Historia: Generación de Reportes		**Riesgo en Desarrollo (Alta/Media/Baja):** Media
Referencia a la Historia de usuario número: 2		**Prioridad en Negocio (Alta/Media/Baja):** Alta
Descripción:	Generar reportes que permita: Ver stock existente por lotes. Despachos realizados. Reportes por periodos de tiempo.	
Observaciones:	Imprimir reportes.	

Tabla2.3.10: Historia de Usuario: Generación Reportes[16]

2.3 ANÁLISIS DE PROCESOS.

Para que el desarrollo de un proyecto de software concluya con éxito, es de suma importancia que antes de empezar a codificar los programas que constituirán la aplicación de software completa, se tenga una completa y plena comprensión de los requisitos del software.

Pressman establece que la tarea del análisis de requisitos es un proceso de descubrimiento, refinamiento, modelado y especificación. Se depura en detalle el ámbito del software, y se crean modelos de los requisitos de datos, flujo de información y control, y del comportamiento operativo. Se analizan soluciones alternativas y se asignan a diferentes elementos del software.

[16] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

El análisis de requisitos permite al desarrollador especificar la función y el rendimiento del software, indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software, éste puede dividirse en cinco áreas de esfuerzo, que son:

1. Reconocimiento del problema. Reconocer los elementos básicos del problema tal y como los perciben los usuarios finales.

2. Evaluación y síntesis. Definir todos los objetos de datos observables externamente, evaluar el flujo y contenido de la información, definir y elaborar todas las funciones del software, entender el comportamiento del software en el contexto de acontecimientos que afectan al sistema.

3. Modelado. Crear modelos del sistema con el fin de entender mejor el flujo de datos y control, el tratamiento funcional y el comportamiento operativo y el contenido de la información.

4. Especificación. Realizar la especificación formal del software.

5. Revisión. Un último chequeo general de todo el proceso.[b]

En el módulo de Gestión de Medicamentos, el análisis de requerimientos debe llevarse a cabo en base a las necesidades administradores y los usuarios, la funcionalidad del software debe emular lo más cercano a la realidad las actividades que se llevan a cabo en la administración de los medicamentos desde el momento que ingresa hasta su despacho.

En el caso de este proyecto el proceso de análisis comenzó con una primera entrevista al Ing. Richard Murillo.

---

[b] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/fuentes_k_jf/capitulo2.pdf

En vista de que este proyecto se plantea como un proyecto de aplicación computacional la parte fundamental de la etapa de análisis es la que se realizó alrededor de las diferentes formas de llevar un ingreso, despacho y bodegaje de la medicación que ingresaba al establecimiento de salud, como todos los procesos que se realizaban para el mismo, que es precisamente lo que se pretende modelar.

Los requerimientos de sistema que se definen en la etapa de análisis de un proceso de Ingeniería de Software generalmente se clasifican como requerimientos funcionales y requerimientos no funcionales. Los principales requerimientos funcionales y no funcionales de la herramienta de software que este proyecto propone se definen en las dos secciones siguientes.

2.3.1 REQUERIMIENTOS FUNCIONALES.

Los requerimientos funcionales son los que se encargan de definir lo que la herramienta de software debe hacer. Definen los alcances del sistema en cuanto a las acciones que debe realizar, y en cuanto a la transferencia de datos entre todas las diferentes funciones del sistema. En el caso de este proyecto, los principales requerimientos funcionales son los siguientes:

- El sistema ofrecerá la posibilidad de llevar un correcto control de los procesos de gestión de medicamentos para garantizar que se los realice de manera transparente y eficaz.

- El sistema ofrecerá la posibilidad de permitir el ingreso, modificación y eliminación de los medicamentos que se ingresan o despachan de bodega.

- El sistema ofrecerá la posibilidad de comprobar la existencia de los medicamentos en bodega antes y después de despachar los pedidos realizados por farmacia.

- El sistema ofrecerá la posibilidad de llevar un control de los medicamentos existentes desplegando reportes: al terminarse el stock de medicamentos que se tiene en bodega, o al estar por fenecer la fecha de caducidad de cada lote de medicamentos.

- El sistema ofrecerá la posibilidad de agilizar el proceso de entrega, inventario, administración y control de medicamentos y de manera simultánea generar resultados en forma rápida y eficiente.

- El sistema ofrecerá la posibilidad de generar reportes de la medicación existente, faltante, y su distribución a las diferentes farmacias o Subcentros de Salud, así como se podrán generar reportes esenciales para el correcto manejo del inventario.

- El sistema ofrecerá la posibilidad de realizar pruebas y ajustes de mantenimiento para que se determine el funcionamiento del sistema.

- El sistema no ofrecerá la posibilidad de realizar la contabilidad de la medicación que ingrese a bodega.

2.3.2 REQUERIMIENTOS NO FUNCIONALES.

Los requerimientos no funcionales hacen relación a las características del sistema que aplican de manera general como un todo, más que a rasgos particulares del mismo y corresponden a aspectos tales como la disponibilidad, mantenibilidad, flexibilidad, seguridad, facilidad de uso, etc., los cuales se describen a continuación:

- **Desempeño:** Garantizar la confiabilidad, la seguridad y el desempeño del sistema de Gestión de Medicamentos, en este sentido la información almacenada podrá ser consultada y actualizada permanente y simultáneamente, sin que se afecte el tiempo de respuesta.

- **Disponibilidad:** Estar disponible 100%, los 365 días del año, las 24 horas del día, con sólo disponer de una conexión a Internet o simplemente con la utilización de la Intranet.

- **Escalabilidad:** El sistema debe ser desarrollado sobre una base de software evolutivo e incremental, que permita en un futuro el desarrollo de nuevas funcionalidades o requerimientos, sin afectar en gran escala el código existente; para ello deben incorporarse aspectos de reutilización de componentes.

- **Facilidad de Uso e Ingreso de Información:** El sistema debe ser de fácil uso, así como de fácil adaptación por parte de los usuarios del Centro de Salud No. 3 “La Tola-Vicentina” de la Dirección Provincial de Salud de Pichincha, además el sistema debe presentar mensajes de error que permitan al usuario identificar el tipo de error y comunicarse con el administrador del sistema.

- **Flexibilidad:** El sistema debe ser diseñado y construido con los mayores niveles de flexibilidad en cuanto a la parametrización de los tipos de datos.

- **Mantenibilidad:** El sistema deberá estar debidamente documentado; tanto en el código fuente como en los manuales de usuario, para que sea más fácil dar mantenimiento al sistema, cuando sea necesario.

2.3.3 DISEÑO:

Según Pressman, el diseño del software es realmente un proceso de muchos pasos pero que se clasifican dentro de uno mismo. En general, la actividad del diseño se refiere al establecimiento de las estructuras de datos, la arquitectura general del software, representaciones de interfaz y algoritmos. El proceso de diseño traduce requisitos en una representación de software.

2.3.4 GENERACIÓN DE CÓDIGO:

Este proceso consiste en traducir el diseño en una forma legible por la máquina. En el caso de la aplicación de software de este proyecto, la generación de código se refiere tanto a la parte de generación de los distintos formularios que conforman el módulo, como a la parte en la cual se añadirá comportamiento a estos a tales formularios. El lenguaje de programación PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica. La interacción con los formularios o distintas interfaces que componen el módulo, es decir, su funcionalidad, se puede construir a través de algún otro lenguaje de programación, como clases Java o scripts especificados en JavaScript, jquerys, AJAX, etc. Todas estas actividades implican generar código.

2.3.5 PRUEBAS:

Una vez que se ha generado código, comienzan las pruebas del software o sistema que se ha desarrollado. De acuerdo con Pressman, el proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de las prueba para la detección de errores. En el caso de una herramienta de

software para el módulo de gestión de medicamentos, es necesario tener etapas de pruebas tanto para la parte funcional del software, como para la parte aplicativa del mismo. Se requiere poder probar el software con las personas que van a manipular la aplicación para que puedan evaluar el comportamiento del software con el fin de proporcionar retroalimentación a los desarrolladores. Es sumamente importante que durante el proceso de desarrollo no se pierda el contacto con los interesados o solicitantes del desarrollo de software, de esta manera los objetivos de proyecto se mantendrán vigentes y se tendrá una idea clara de los aspectos que tienen que probarse durante el periodo de pruebas.

2.3.6 MANTENIMIENTO:

El software indudablemente sufrirá cambios, y habrá que hacer algunas modificaciones a su funcionalidad. Es de suma importancia que el software de calidad pueda adaptarse con fines de acoplarse a los cambios de su entorno externo, por medio de la documentación apropiada y atinada del software se pueden presentar las vías para el mantenimiento y modificaciones al mismo. En el capítulo 4, Desarrollo, se describen los pasos para la construcción y desarrollo de este módulo, de tal manera que este mismo documento puede utilizarse posteriormente como referencia al cómo deben desarrollarse aplicaciones que sigan la metodología de trabajo que este proyecto propone y así se pueda lograr crear políticas que encajen en el buen funcionamiento del software sino también en la institución a ser implementado.

2.4 ANÁLISIS DE LOS REQUISITOS DEL SOFTWARE.

El proceso de reunión de requisitos se intensifica y se centra especialmente en el software. Dentro del proceso de análisis es fundamental que a través de una colección de requerimientos funcionales

y no funcionales, el desarrollador del software comprenda completamente la naturaleza de los programas que deben construirse para desarrollar la aplicación, la función requerida, comportamiento, rendimiento e interconexión. En el caso de este proyecto, el proceso de análisis y de obtención de requerimientos se lleva cabo a través de trabajar conjuntamente con el Centro de Salud No. 3 La Tola – Vicentina bajo la tutela de la Dirección Provincial de Salud de Pichincha, quienes proporcionan los parámetros bajo los cuales la aplicación debe desarrollarse para poder de esta manera cumplir con los objetivos de este proyecto.

2.4.1 INFORMÁTICA MÉDICA:

Informática médica es la aplicación de la informática y las comunicaciones al área de la salud, mediante el uso del software médico formando parte de las tecnologías sanitarias. Su objetivo principal es prestar servicio a los profesionales de la salud para mejorar la calidad de la atención a la salud; esta es la principal función para la realización del presente módulo facilitar la labor de los funcionarios de la salud así poder brindar una atención eficiente a los usuarios de los servicios de bodega con la ocupación de recursos, dispositivos y métodos necesarios para optimizar la adquisición, almacenamiento, recuperación y utilización de la información en salud. Los instrumentos informáticos de la salud incluyen no sólo los ordenadores, sino también guías de práctica clínica, terminología médica formal, y de sistemas de información y comunicación.

Engloba a varias subdisciplinas, algunas de las más importantes son:

- Informática Médica de orientación clínica
- Informática aplicada a Salud Pública y Epidemiología
- Telemedicina
- Informática aplicada a la Enfermería

- Información al paciente y consumidores

Tiene aplicación en todas las áreas de la medicina, como en laboratorios de análisis clínicos, dispositivos electrónicos para hacer mediciones, PACS (siglas en inglés que su significado en español es de "*sistema de archivado y transmisión de imágenes*"), software de gestión hospitalaria, de manejo de turnos, de historias clínicas, bases de datos de pacientes, entre otros.

Por tal motivo, la Informática Médica es un campo multidisciplinario que acoge a profesionales de áreas como la biomedicina, informática de sistemas, telecomunicaciones, electrónica, administración y gestión, etc.

2.4.1.1 Aspectos de la esfera

Sistemas de apoyo de decisiones en la asistencia sanitaria, incluidos los sistemas de apoyo de decisiones clínicas. Arquitectura de registros médicos electrónicos y otros sistemas de información utilizados para la facturación, inventario y recopilación de información de los medicamentos existentes en cada centro de salud.

Normas (por ejemplo, DICOM y HL7) y la integración de perfiles (por ejemplo, la integración de los Servicios Médicos de Empresa), para facilitar el intercambio de información entre los sistemas de información de salud - específicamente y definir los medios para el intercambio de datos, no el contenido.

Es considerada como una especialidad médica, ya que a pesar de tener un componente tecnológico de informática e ingeniería, el estudio del conocimiento médico y su aplicación en el manejo del paciente a través de sistemas de información y telecomunicaciones,

requiere de una extensa base de formación médica, clínica y de las posibilidades de la tecnología de distribuir esta información en forma rápida y efectiva.

No debe confundirse con la informática administrativa en salud que no se reconoce tradicionalmente como informática médica, ya que no trata directamente con el cuidado de la salud sino con la administración de recursos.

2.4.1.2 Utilidad Pública

El problema que tiene el sistema de atención de la salud, es que está fuera de control, tanto en el registro de datos, la práctica médica y el software que no se registran encuentros clínicos.
En lugar de ello, los registros médicos electrónicos y las empresas de software no son capaces de comunicar a través de sus plataformas de software provincial.

La solución a este grave problema es simple realmente.

La Informática Médica de Servicios Públicos serviría de "plataforma común" de la comunicación para todos los productos de software existentes de provincia, así como el seguro de depósito para el público de los registros médicos.

El potencial para la reducción de los errores médicos, el fraude y la reducción de la duplicación es asombroso. El número de vidas salvadas podría superar un mínimo de 100 mil por año, según el Instituto de Medicina actual del error médico estadísticas de mortalidad.

2.4.2 IDENTIFICACIÓN DE LOS ELEMENTOS DEL MÓDULO DE GESTIÓN DE MEDICAMENTOS.

La identificación de cada uno de estos elementos tiene un papel sumamente importante en las etapas de análisis y diseño de la herramienta aplicativa que este proyecto propone. Estos elementos son los siguientes:

- Usuarios. Son aquellos que van a interactuar con el sistema de Gestión de Medicamentos. En el caso de este proyecto, se refiere a aquellas personas que se encargan de la Administración de la parte de medicamentos como su ingreso, bodegaje y posterior entrega teniendo un control eficiente de cada uno de esos procesos.

- Interfaz Gráfica. Es la vista que se presenta al usuario. Para este proyecto, se establece la interfaz que se presentará al usuario dependiendo de su perfil y del grado de responsabilidad que tenga en la administración del sistema.

- Especialistas. Se refiere al manejo del sistema, generalmente el grupo de especialistas consiste en personas con diferentes especialidades, desde el médico hasta los programadores y diseñadores del módulo.

- Acceso, infraestructura y conectividad. Este elemento se refiere a la arquitectura general que el módulo va a presentar para su desarrollo. Como ya se ha mencionado anteriormente, este proyecto se desarrolla principalmente en PHP con el fin de aprovechar todas las ventajas de portabilidad que presenta Internet.

2.5 GENERACIÓN DE ENTRADAS Y SALIDAS DEL SISTEMA.

Las entradas y salidas de un sistema informático es el enlace que une al sistema con el mundo y sus usuarios, en esta existen aspectos generales que todos los analistas deben tener en cuenta estos son:

2.5.1 REQUERIMIENTOS DE ENTRADA:

Es el desarrollo de especificaciones y procedimientos para la preparación de datos, el cual nos ayuda a poner los datos de transacción en una forma utilizable para su procesamiento.

Existen cinco objetivos que controlan la cantidad de entrada requerida:

- Control de la calidad de entrada. Las Operaciones de preparación y entrada dependen de las personas, dado que los costos de mano de obra son altos y la preparación de ingreso de los datos también lo son.

- Evitar los retrasos. También conocido con el nombre de cuello de botella son siempre uno de los objetivos que el analista evita al diseñar la entrada, una forma de evitarle es utilizar los documentos de retorno.

- Evitar los errores en los datos. La tasa de errores depende de la cantidad de datos, ya que entre más pequeña sea esta menores serán las oportunidades para cometer errores.

- Evitar los pasos adicionales. Evitar diseños para la entrada que traigan una mayor cantidad de pasos a seguir.

- Mantener la sencillez del proceso. El sistema mejor diseñado se ajusta a las personas que lo utilizarán y al mismo tiempo

proporcionarán métodos para el control de los errores, la simplicidad funciona y es aceptada por cualquier usuario.

2.5.1.1 Actividades de entrada que realiza el Sistema:

Acceso a los Datos generales de la nómina de Usuarios de la Dirección provincial de salud, por parte del módulo de Recursos Humanos, el cual nos proporcionará la información necesaria de cada uno de los responsables de las diferentes áreas de la Dirección.

2.5.2 REQUERIMIENTOS DE SALIDA:

El diseño de sistema se representa a través de dos fases: el diseño lógico y el diseño físico.

- El diseño lógico. Especifica las formas de entrada y las descripciones de las pantallas de todas las transacciones y archivos a fin de mantener los datos de inventario, los detalles de las transacciones y los datos del proveedor. Las especificaciones de los procedimientos describen métodos para introducir los datos, corridas de informes copiados de archivos y detección de problemas.

- El diseño físico. Actividad que sigue el diseño lógico, produce programas de software, archivos y un sistema en marcha, las especificaciones del diseño indican a los programadores que debe hacer el sistema. Los programadores a su vez escriben los programas que aceptan entradas por parte de los usuarios, procesan los datos, producen los informes y almacenan estos datos en los archivos.

El alcance del diseño de sistemas se guía por el marco de referencia para el nuevo sistema desarrollado durante el análisis. Los datos de los requerimientos, recopilados durante la investigación, conforman las

actividades y componentes del sistema. Se identifican las características generales, como informes y entradas; en el diseño de la salida por ejemplo, los analistas deben conocer la longitud de campo de un dato específico para establecer cuanto espacio dejar en la información, en la pantalla de despliegue visual o archivo.

Además, la participación del usuario proporciona al analista una retroalimentación importante conforme avanza en el diseño; además asegura a los usuarios tengan un conocimiento no técnico de lo que realizará o no el nuevo sistema.

Los requerimientos del sistema y las especificaciones de diseño se establecen con claridad y son muy bien entendidas, y los analistas tienen la experiencia para convertir los requerimientos en un sistema eficiente y que trabaje bien.

A menudo, para los usuarios la característica más importante de un sistema de información es la salida que produce. Si la salida no es de calidad, se pueden convencer de que todo el sistema es tan innecesario que eviten su utilización y, por lo tanto, posiblemente ocasionen errores y que el sistema falle. Él termino “salida” se aplica a cualquier información producida por un sistema, ya sea impresa, desplegada o verbal. Cuando los analistas diseñan la salida, seleccionan métodos para representar la información y crean documentos, informes u otros formatos que contienen información producida por el sistema.

Los métodos de salida varían a lo largo de los sistemas.
Los analistas deben decidir cuándo imprimir, desplegar o presentar su salida en forma audible.

Los sistemas de información ya sean que se desarrollen sobre sistemas pequeños de escritorio o sobre grandes sistemas, utilizan 3 métodos principales para la salida los cuales se clasifican en:

- Impresión
- Pantalla
- Despliegue y audio

2.5.2.1 Actividades de salida que realiza el Sistema:

- Generar reportes dependiendo de los privilegios que tenga cada usuario en el sistema, e imprimirlos.
- Acceso a los datos de la tabla producto por parte del personal autorizado de los módulos de farmacia y vacunas.
- Imprimir los despachos de medicamentos que se han realizado.
- Señalar eventos importantes, problemas ó advertencia que ocurra en cuanto a la medicación.

3. CAPÍTULO III – DISEÑO

3.1 DIAGRAMAS DE CASOS DE USOS.

Para desarrollar diagramas de caso de uso, es importante saber cada una de las actividades o pasos que se van a ir desarrollando desde el primer momento hasta el momento final del sistema.

Esto nos ayuda a comprender la forma en que el sistema deberá comportarse, obteniendo los requerimientos desde el punto de vista del usuario.

Los elementos que se emplean:

- Actor, que es quien inicia, y luego otro actor (que puede ser el mismo que inicia el caso de uso o puede ser otro diferente), que recibirá algo por parte del sistema. El actor que inicia se encuentra a la izquierda del caso de uso, y el que recibe a la derecha. El nombre del actor aparece justo debajo de él, y el nombre del caso de uso aparece ya sea dentro de la elipse o justo debajo de ella.

- La elipse representa el caso de uso.

- Una línea asociativa conecta a un actor con el caso de uso, y representa la comunicación entre el actor y el caso de uso.

3.1.1 CASO DE USO INICIAR SESIÓN.

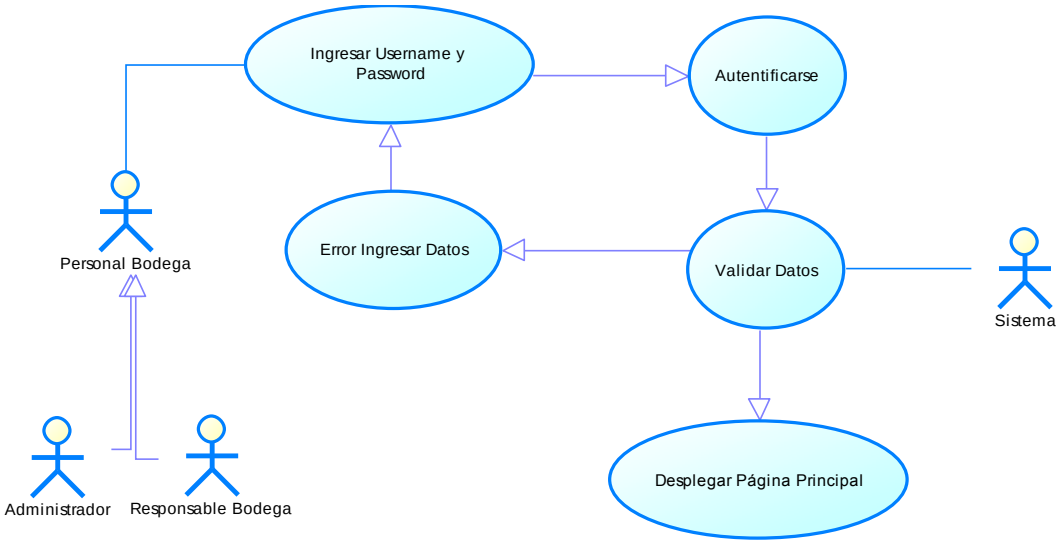


Figura 3.1.1. Caso de uso Iniciar Sesión[17]

3.1.1.1 Especificación de Caso de Uso Iniciar Sesión.

Nombre	Iniciar Sesión
Objetivo	Ingresar al Sistema SGMAS Gestión de Medicamentos.
Actores	Personal de Bodega. Sistema (acciones que realiza el módulo dependiendo de las peticiones de los usuarios)
Precondiciones	Para que el usuario pueda ingresar al sistema es necesario que conozca su usuario (username) y contraseña (password), previamente establecidos.
Acciones Básicas	• **Ingresar al Sistema.** • **Ingresar username y password.** • El sistema valida los datos ingresados. • Se despliega la pantalla inicial de acuerdo al perfil de usuario.
Post condiciones	Los datos ingresados correctamente dan acceso al sistema.

Tabla 3.1.1.1 Especificación de caso de uso de Inicio de Sesión [18]

[17] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.1.2 CASO DE USO GENERAR ORDEN DE DESPACHO

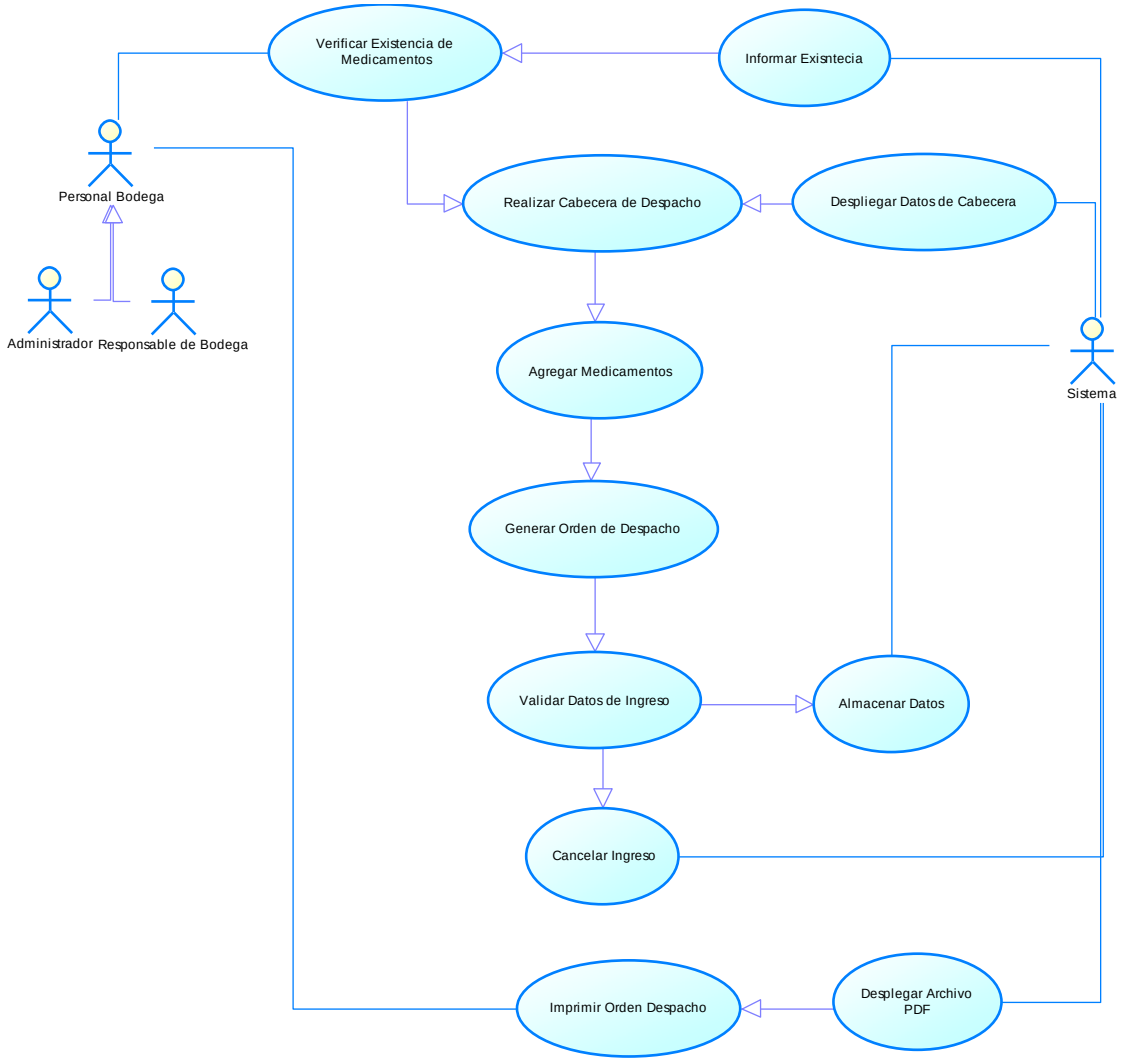


Figura 3.1.2. Caso de Uso Generar Orden de Despacho[19]

[18] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*
[19] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.1.2.1 Especificación de Caso Uso Generar Orden de Despacho.

Nombre	Generar Orden de Despacho
Objetivo	Generar la Orden de Despacho de los medicamentos a farmacia.
Actores	Personal de Bodega. Sistema (acciones que realiza el módulo dependiendo de las peticiones de los usuarios)
Precondiciones	El Responsable de Bodega debe tener la orden de pedido generada por farmacia. Consultar si existe stock disponible de medicamentos en bodega.
Acciones Básicas	• Verificar Existencia de Medicamentos: Consultar la existencia de medicamentos que se van a despachar. • Realizar Cabecera de Despacho: El Responsable de Bodega para realizar un despacho, debe tener realizada una cabecera, que comprende los siguientes datos: (Nombre de la bodega, Transacción, Destinatario, Fecha Movimiento, Estado Movimiento, Núm. Referencia). • Agregar Medicamentos a la Orden de Despacho: El Responsable de Bodega realiza una búsqueda de los medicamentos que se van a despachar, (en caso de no existir dicho medicamento se debe ingresar primero los

	datos del mismo), se asigna la cantidad de cada uno. • Identificar Tipo de Programa por Medicamento: Se debe identificar el tipo de programa por medicamento sea este por Maternidad o Fiscal. • Generar Orden de Despacho: Una vez ingresados todos los datos el Responsable de bodega genera la orden de despacho, revisando que los datos estén correctamente ingresados. • Imprimir Orden de Despacho: El Responsable de bodega imprime la Orden de despacho siempre eligiendo el tipo de programa por medicamento que se va a despachar. • Entregar Pedido: El Responsable de Bodegaentrega la orden de despacho, y los medicamentos a farmacia.
	El sistema genera un comprobante de despacho.
Post condiciones	Farmacia recepta los productos, y verifica pedido.

Tabla 3.1.2.1 Especificación de Caso Uso Orden de Despacho.[20]

[20] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.1.3 CASO DE USO DE INGRESAR LOTES DE MEDICAMENTOS.

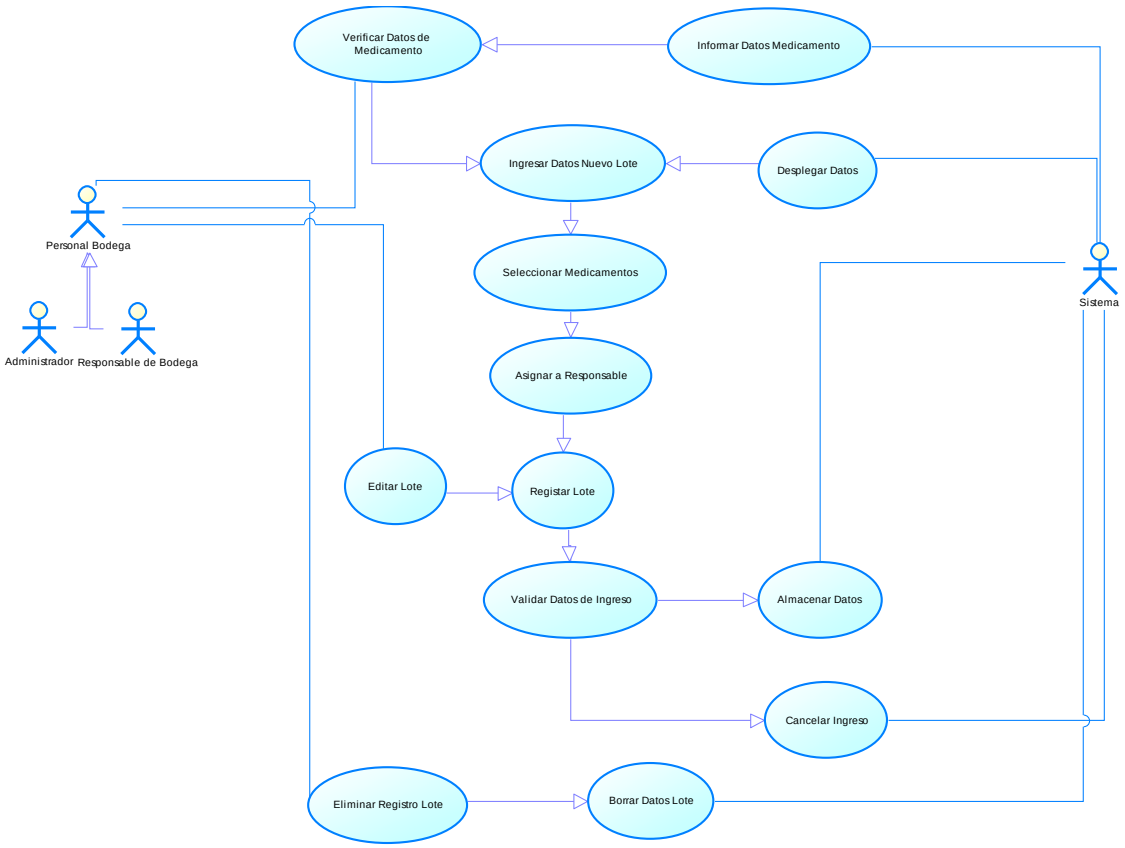


Figura.3.1.3 Caso de Uso Ingresar Lotes de Medicamentos.[21]

3.1.3.1 Especificación de Caso Uso Ingresar Lotes de Medicamentos.

Nombre	Ingresar Lotes de Medicamentos.

[21] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Objetivo	Ingresar Lotes de Medicamentos al sistema para poder llevar un control adecuado y eficiente de la medicación existente.
Actores	Personal de Bodega. Sistema (acciones que realiza el módulo dependiendo de las peticiones de los usuarios)
Precondiciones	El proveedor entrega los medicamentos al Responsable de Bodega para ser almacenados por los lotes de Medicamentos. Para poder almacenar un nuevo lote de medicamentos, deben estar primeramente registrados los datos de los productos que se van a ingresar, a demás el Núm. De Documento Contable, y el Tipo de Programa al que pertenece.
Acciones Básicas	• Ingresar Nuevo Lote: Para poder almacenar un nuevo lote de medicamentos, deben estar registrados los datos del medicamento que se va a ingresar, teniendo esos datos disponibles el Responsable de Bodega puede ingresar un nuevo Lote a Bodega. Ingresa en Núm. de Lote. • Seleccionar Medicamentos: El Responsable de Bodega selecciona los medicamentos que se van a ingresar, (de no existir el producto se debe ingresar los datos generales del mismo). • Seleccionar Número Documento Contable:

	Seleccionar el número de documento contable en caso de no estar registrado, se debe realizar el ingreso del Documento Contable. • Ingresar datos Recepción: El Responsable de Bodega ingresa: Responsable, Transacción, el tipo de programa, y también si tiene alguna observación. • Registrar Nuevo Lote Bodega: El Responsable de Bodega verifica los datos ingresados, guarda la Orden de Ingreso de Productos y se actualiza la base de datos con el nuevo stock.
Post condiciones	Los medicamentos están almacenados por número de lotes, y actualizando al stock general.

Tabla 3.1.3.1Especificación de Caso Uso Ingresar Lotes de Medicamentos.[22]

3.1.4 CASO DE USO DE REGISTRAR MEDICAMENTO.

[22] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

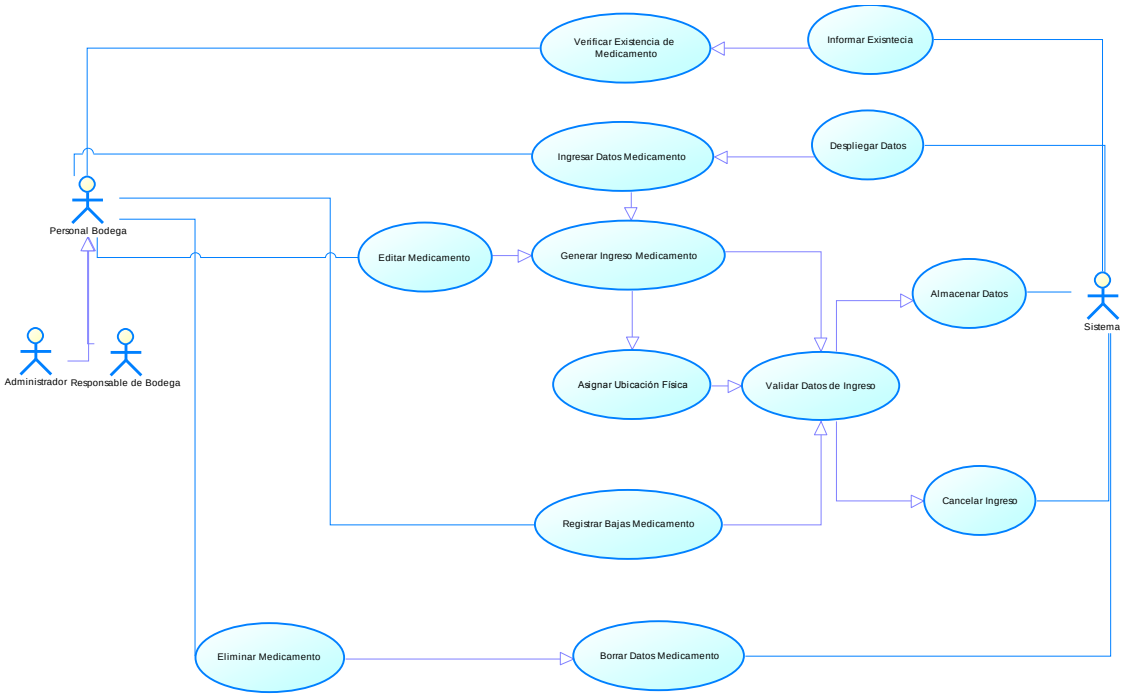


Figura.3.1.4 Caso de Uso Registrar Medicamento[23]

3.1.4.1 Especificación de Caso Uso Registrar Medicamento.

Nombre	Registrar Medicamento
Objetivo	Ingresar un nuevo Medicamento al Sistema.
Actores	Personal de Bodega. Sistema (acciones que realiza el módulo dependiendo de las peticiones de los usuarios)
Precondiciones	Para realizar el ingreso de un nuevo medicamento, se debe tener el registro de la siguiente información: • Nombre Bodega • Fórmula Farmacéutica • Familia: dependiendo si es insumos médicos, medicamentos, etc. • Concentración: Unidad de medida de los

[23] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

	medicamentos (250 MG, 600 MG, etc.) • Presentación Comercial: Presentación del medicamento (crema, granulado, jalea, etc.)
	• Ingresar Datos Medicamentos: El Responsable de Bodega ingresa los datos generales del medicamento: el código CUM asignado a ese medicamento.
	• Escoger Bodega: Se selecciona el nombre de la bodega a la cual va asignada el medicamento.
	• Escoger Fórmula Farmacéutica: Se selecciona la fórmula Farmacéutica del medicamento.
Acciones Básicas	• Ingresar Nombre del Producto: Se Ingresa el nombre del medicamento.
	• Escoger Estado del Producto. Se selecciona el estado del medicamento.
	• Escoger Familia: Se selecciona la familia a la cual pertenece el medicamento.
	• Escoger Concentración: Se selecciona la concentración del medicamento.
	• Escoger Presentación Comercial: Se selecciona la presentación comercial a la cual pertenece el medicamento.

	• Ingresar Stock Máximo y Stock Mínimo del medicamento: Se ingresa el stock máximo y mínimo del medicamento. • Registrar Producto: Se almacena los datos del medicamento.
Post condiciones	El ingresar máximos y mínimos de stock, permite llevar un control más eficiente de la medicación, ya que el momento que se genere los reportes se podrá observar cuando un medicamento esta por agotarse o al contrario si existe demasiado de alguno de ellos.

Tabla 3.1.4.1 Especificación de Caso Uso Registrar Medicamento.[24]

3.1.4.2 Especificación de Caso Uso Asignar Ubicación Física.

Nombre	Asignar Ubicación Física
Objetivo	Ingresar la ubicación física de los medicamentos, para poder facilitar la distribución y búsqueda en el momento de despachar o buscar alguna medicación.
Actores	Personal de Bodega
Precondiciones	Deben estar registrados los medicamentos con anterioridad.
Acciones Básicas	El Responsable de Bodega asignará la ubicación física de los medicamentos.

[24] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

	Se ingresa: sector, percha, fila.
Post condiciones	Mejor distribución de los lotes de medicación.

Tabla 3.1.4.2 Especificación de Caso Uso Asignar Ubicación Física[25]

3.1.4.2 Especificación de Caso Uso Registrar Bajas de Medicamentos.

Nombre	Registrar Bajas de Medicamentos.
Objetivo	Permitir llevar un control de los productos que han sido dados de baja especificando el motivo.
Actores	Personal de Bodega
Precondiciones	Deben estar registrados los medicamentos con anterioridad. Tener los permisos de usuario para poder realizar esta acción.
Acciones Básicas	El Responsable de Bodega ingresa el medicamento a darse de baja y especifica los motivos por lo cual se realiza esta acción.
Post condiciones	Se actualiza el stock de medicamentos. Registro de medicamentos y causas por las cuales fueron dados de baja.

Tabla 3.1.4.3 Especificación de Caso Uso Registrar Bajas de Medicamentos.[26]

3.2 DIAGRAMA DE SECUENCIAS.

El diagrama de secuencias muestra los objetos participando en la interacción y la secuencia de mensajes intercambiados, contiene:

- Objetos con sus “líneas de vida”
- Mensajes intercambiados entre objetos en una secuencia ordenada
- Línea de Vida Activa (opcional)

El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y los actores participantes en la interacción, sin un orden prefijado,

[25] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*
[26] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

aunque por orden lo usual es colocar los objetos de izquierda a derecha y en la parte superior.

Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. Se pueden colocar etiquetas (como restricciones del tiempo, descriptores de acciones, etc.) bien el margen izquierdo o bien junto a las transiciones o a las activaciones a las que se refieren.

Cada línea de vida de un objeto es una línea discontinua que se desplaza hacia abajo del objeto. Una línea continua con una punta de flecha conecta una línea de vida con otra, y representa un mensaje de un objeto a otro. El tiempo se inicia en la parte superior y continúa hacia abajo. Aunque un actor es el que **normalmente inicia una secuencia, su símbolo no es parte de conjunto de** símbolos del diagrama de secuencias.

Un mensaje puede ser simple, síncrono o asíncrono.

- **Mensaje simple** es la transferencia del control de un objeto a otro.
- **Mensaje síncrono** es aquel en el que el objeto espera la respuesta a ese mensaje antes de continuar con su trabajo.
- **Mensaje asíncrono** es aquel en el que el objeto no espera la respuesta a ese mensaje antes de continuar.

No necesariamente se debe especificar el tiempo de manera explícita (como en segundos, minutos, horas, días, etc.), aunque se puede hacer si resulta necesario o conveniente. En todo caso lo que siempre se tiene que hacer ubicar correctamente en el eje vertical la secuencia correcta de eventos que se deben ir dando de forma cronológica o en una línea de tiempo.

3.2.1 DIAGRAMA DE SECUENCIA SGMAS:

3.2.1.1 Diagrama de Secuencia Administrador de Sistema.

Figura.3.2.1.1 Diagrama de Secuencia Administrador de Sistema.

3.2.1.2 Diagrama de Secuencia Ingreso de Nuevo Producto.

Figura.3.2.1.2. Diagrama de Secuencia Ingreso de Nuevo Producto.

3.2.1.3 Diagrama de Secuencia Ingreso Lotes de Productos.

Figura.3.2.1.3. Diagrama de Secuencia Ingreso Lotes de Productos.

3.2.1.4 Diagrama de Secuencia Orden de Despacho.

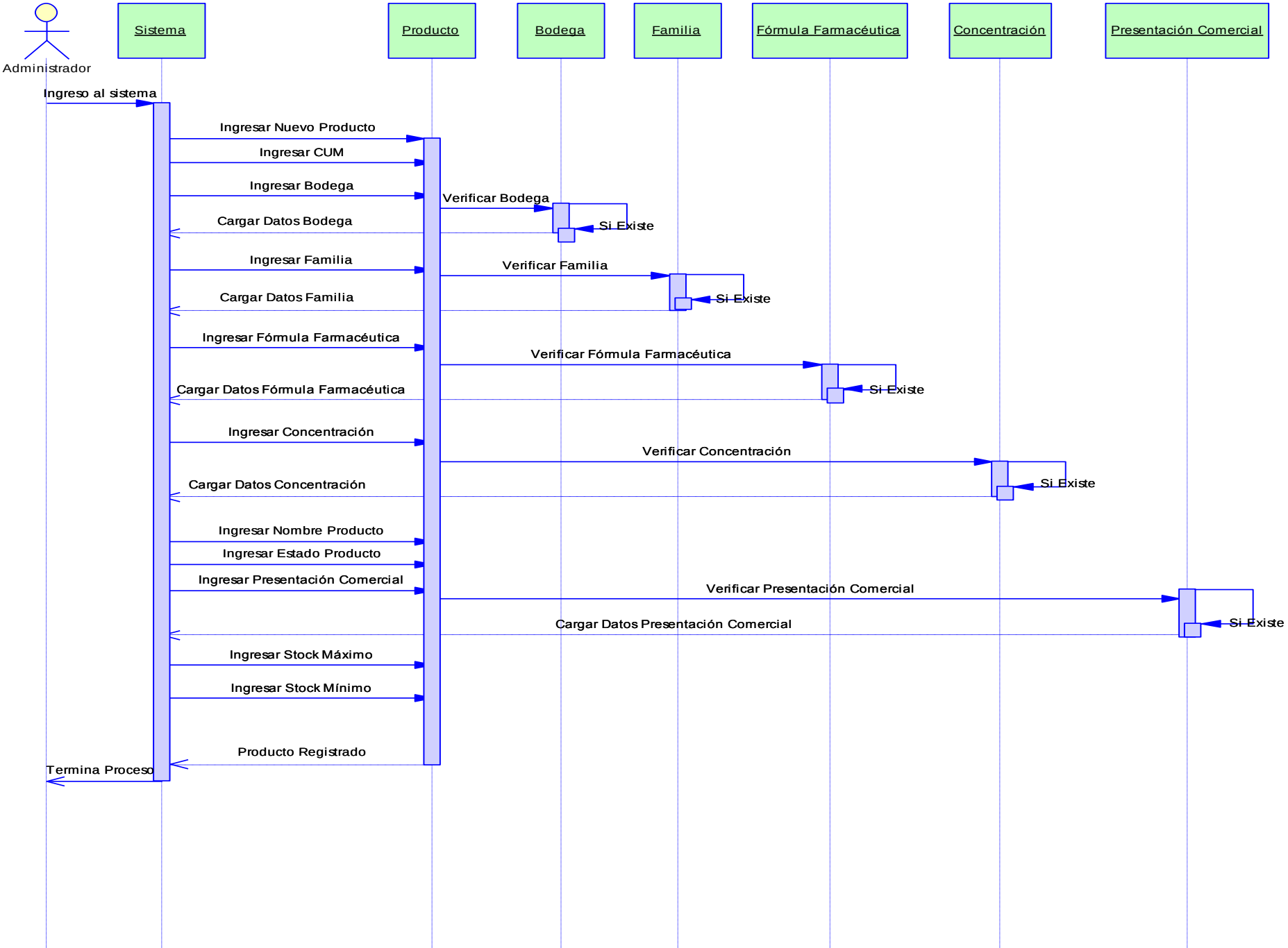
Figura.3.2.1.4. Diagrama de Secuencia Orden de Despacho.

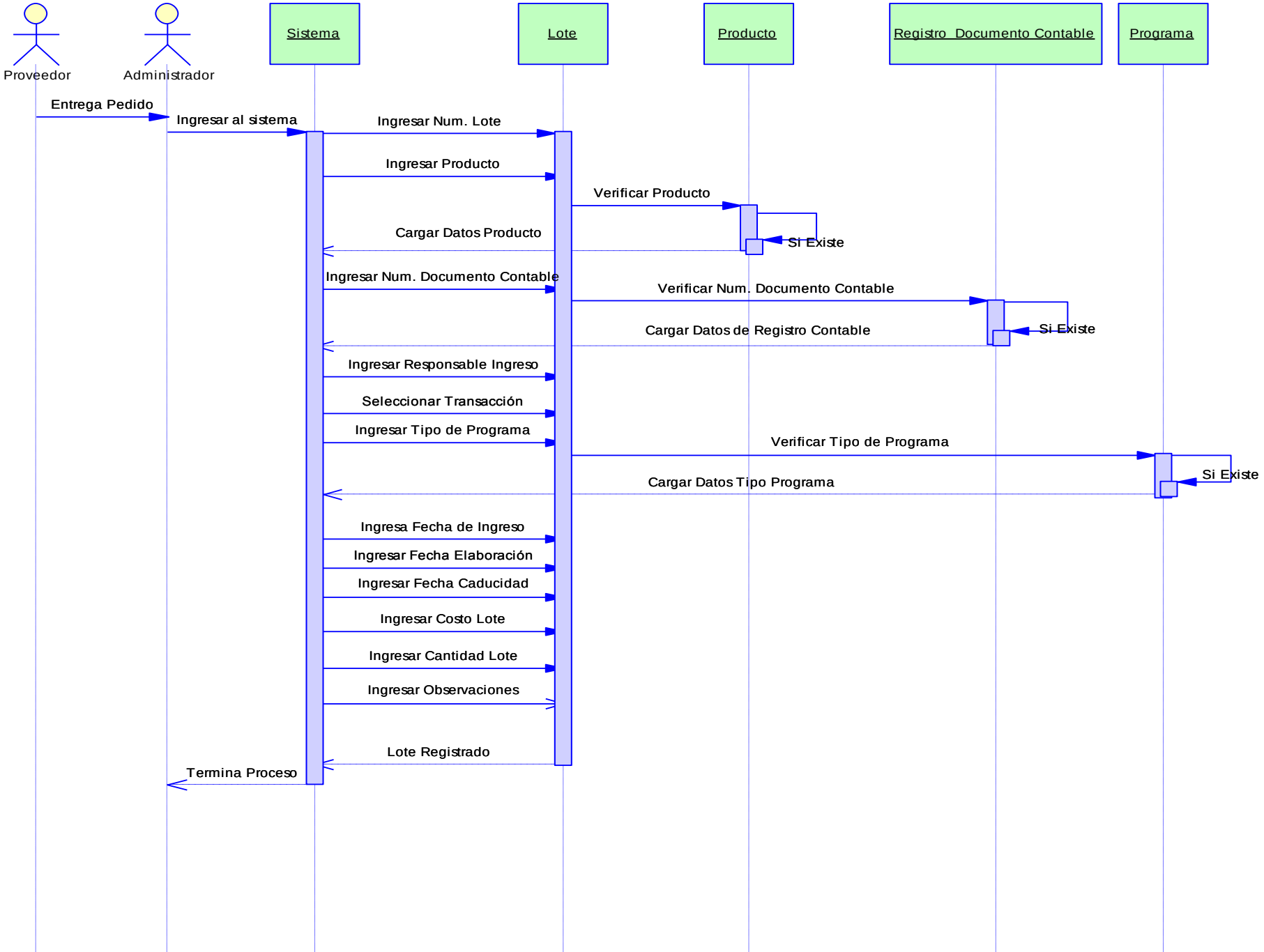
3.2.1.5 Diagrama de Secuencia Registro de Documento Contable

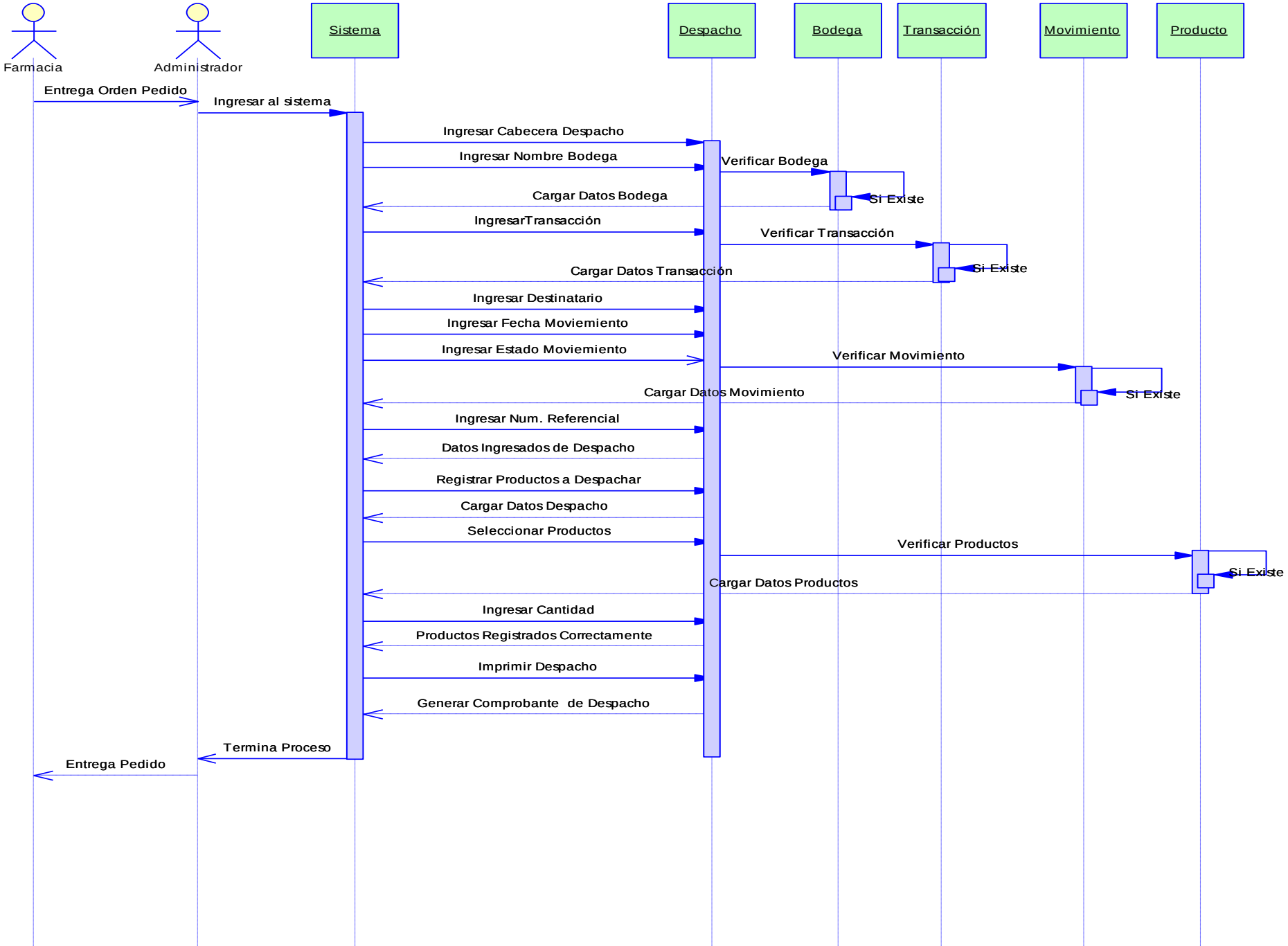
Figura.3.2.1.5. Diagrama de Secuencia Registro de Documento Contable

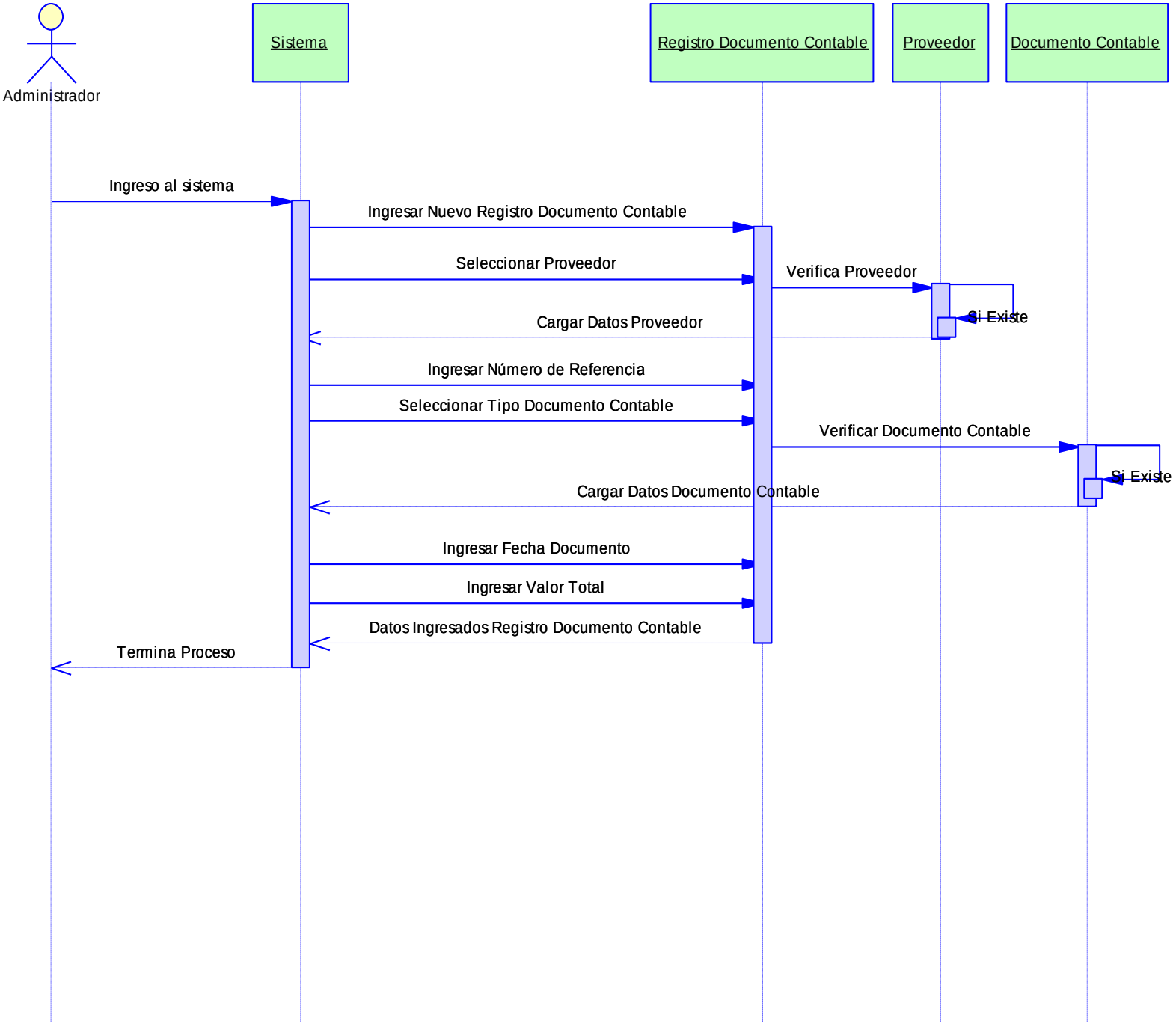
3.2.1.6 Diagrama de Secuencia Nuevo Responsable.

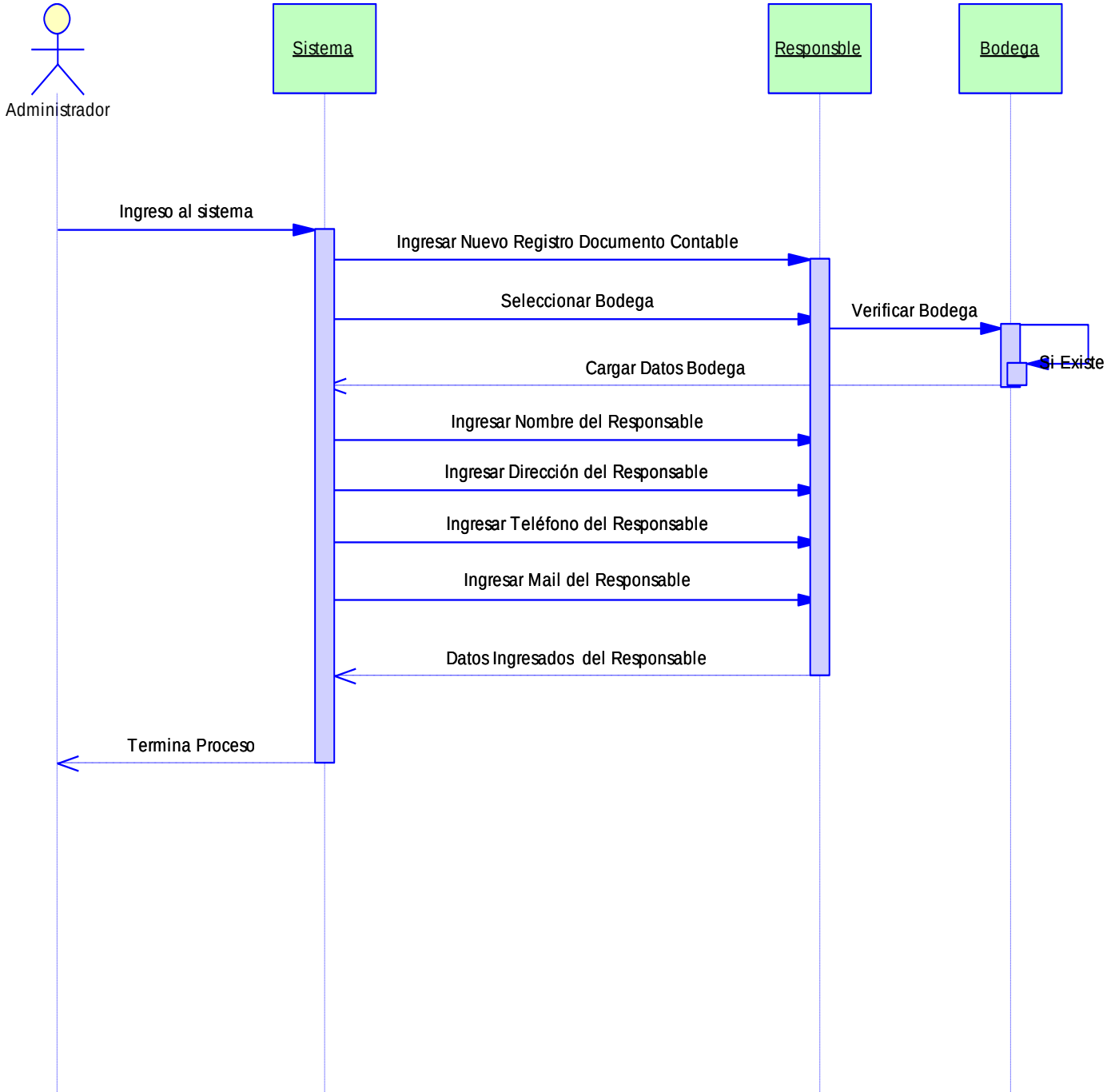
Figura.3.2.1.6. Diagrama de Secuencia Nuevo Responsable.

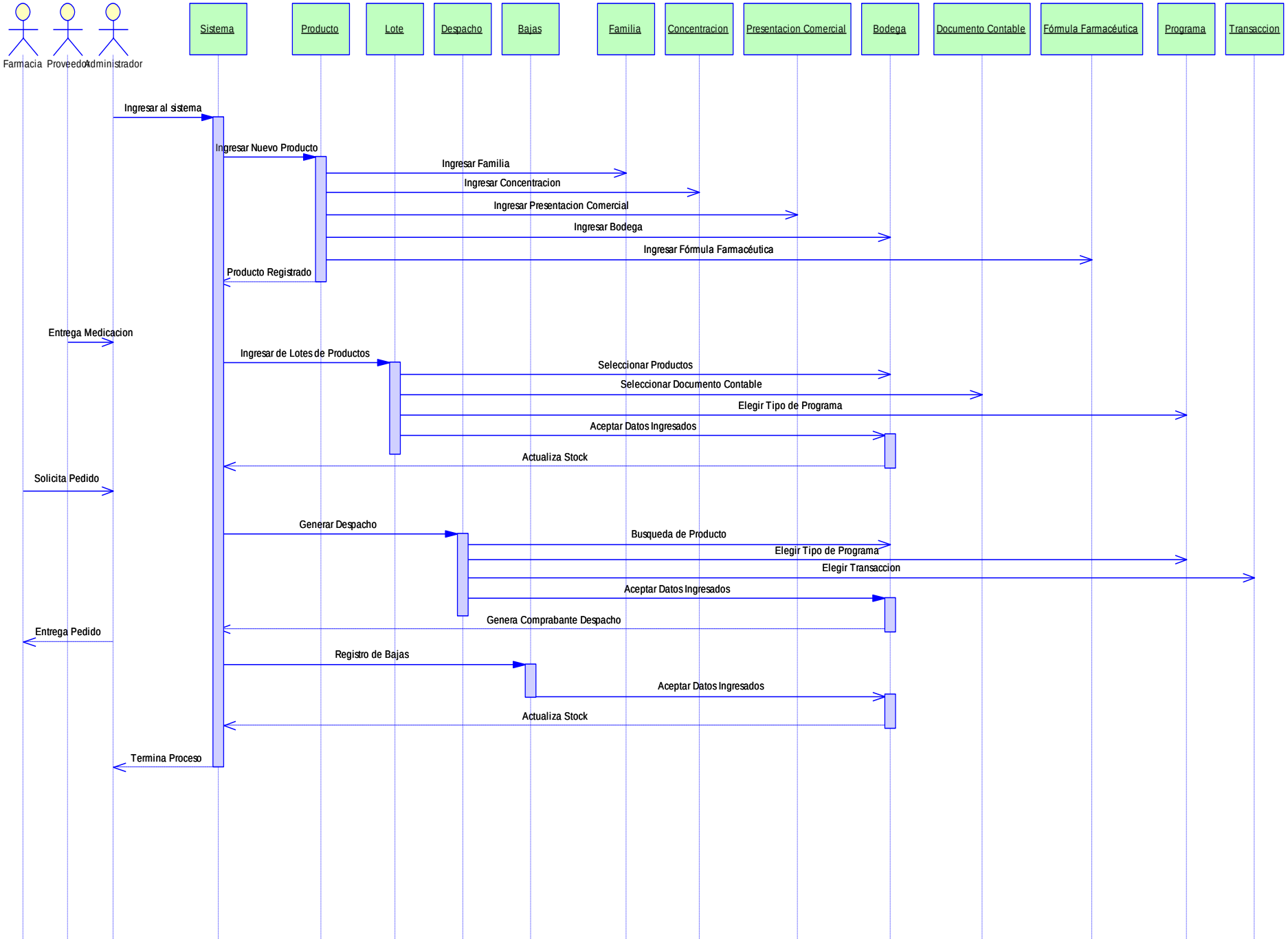

Administrador
Sistema
Producto
Bodega
Familia
Fórmula Farmacéutica
Concentración
Presentación Comercial
Ingreso al sistema
Ingresar Nuevo Producto
Ingresar CUM
Ingresar Bodega
Verificar Bodega
Si Existe
Cargar Datos Bodega
Ingresar Familia
Verificar Familia
Si Existe
Cargar Datos Familia
Ingresar Fórmula Farmacéutica
Verificar Fórmula Farmacéutica
Si Existe
Cargar Datos Fórmula Farmacéutica
Ingresar Concentración
Verificar Concentración
Si Existe
Cargar Datos Concentración
Ingresar Nombre Producto
Ingresar Estado Producto
Ingresar Presentación Comercial
Verificar Presentación Comercial
Si Existe
Cargar Datos Presentación Comercial
Ingresar Stock Máximo
Ingresar Stock Mínimo
Producto Registrado
Termina Proceso


Proveedor
Administrador
Sistema
Lote
Producto
Registro Documento Contable
Programa
Entrega Pedido
Ingresar al sistema
Ingresar Num. Lote
Ingresar Producto
Verificar Producto
Si Existe
Cargar Datos Producto
Ingresar Num. Documento Contable
Verificar Num. Documento Contable
Si Existe
Cargar Datos de Registro Contable
Ingresar Responsable Ingreso
Seleccionar Transacción
Ingresar Tipo de Programa
Verificar Tipo de Programa
Si Existe
Cargar Datos Tipo Programa
Ingresa Fecha de Ingreso
Ingresar Fecha Elaboración
Ingresar Fecha Caducidad
Ingresar Costo Lote
Ingresar Cantidad Lote
Ingresar Observaciones
Lote Registrado
Termina Proceso


Farmacia
Administrador
Sistema
Despacho
Bodega
Transacción
Movimiento
Producto
Entrega Orden Pedido
Ingresar al sistema
Ingresar Cabecera Despacho
Ingresar Nombre Bodega
Verificar Bodega
Si Existe
Cargar Datos Bodega
IngresarTransacción
Verificar Transacción
Si Existe
Cargar Datos Transacción
Ingresar Destinatario
Ingresar Fecha Moviemiento
Ingresar Estado Moviemiento
Verificar Movimiento
Si Existe
Cargar Datos Movimiento
Ingresar Num. Referencial
Datos Ingresados de Despacho
Registrar Productos a Despachar
Cargar Datos Despacho
Seleccionar Productos
Verificar Productos
Si Existe
Cargar Datos Productos
Ingresar Cantidad
Productos Registrados Correctamente
Imprimir Despacho
Generar Comprobante de Despacho
Termina Proceso
Entrega Pedido


Administrador
Sistema
Registro Documento Contable
Proveedor
Documento Contable
Ingreso al sistema
Ingresar Nuevo Registro Documento Contable
Seleccionar Proveedor
Verifica Proveedor
Si Existe
Cargar Datos Proveedor
Ingresar Número de Referencia
Seleccionar Tipo Documento Contable
Verificar Documento Contable
Si Existe
Cargar Datos Documento Contable
Ingresar Fecha Documento
Ingresar Valor Total
Datos Ingresados Registro Documento Contable
Termina Proceso


Administrador
Sistema
Responsble
Bodega
Ingreso al sistema
Ingresar Nuevo Registro Documento Contable
Seleccionar Bodega
Verificar Bodega
Si Existe
Cargar Datos Bodega
Ingresar Nombre del Responsable
Ingresar Dirección del Responsable
Ingresar Teléfono del Responsable
Ingresar Mail del Responsable
Datos Ingresados del Responsable
Termina Proceso


Farmacia
Proveedor
Administrador
Sistema
Producto
Lote
Despacho
Bajas
Familia
Concentracion
Presentacion Comercial
Bodega
Documento Contable
Fórmula Farmacéutica
Programa
Transaccion
Ingresar al sistema
Ingresar Nuevo Producto
Ingresar Familia
Ingresar Concentracion
Ingresar Presentacion Comercial
Ingresar Bodega
Ingresar Fórmula Farmacéutica
Producto Registrado
Entrega Medicacion
Ingresar de Lotes de Productos
Seleccionar Productos
Seleccionar Documento Contable
Elegir Tipo de Programa
Aceptar Datos Ingresados
Actualiza Stock
Solicita Pedido
Generar Despacho
Busqueda de Producto
Elegir Tipo de Programa
Elegir Transaccion
Aceptar Datos Ingresados
Genera Comprabante Despacho
Entrega Pedido
Registro de Bajas
Aceptar Datos Ingresados
Actualiza Stock
Termina Proceso

3.3 DIAGRAMA DE ESTADO

Conforme un sistema interactúa con los usuarios y (posiblemente) con otros sistemas, los objetos que lo conforman pasan por cambios necesarios para ajustar las interacciones. Por esa razón se necesita contar con un mecanismo para cambios en el modelo. Un cambio en un sistema se da debido a que los objetos que componen dicho sistema modificaron su estado como respuesta a los sucesos y al tiempo. Un diagrama de estados se conoce también como un “motor de estado”.

Describen todos los estados posibles en los que puede entrar un objeto particular y la manera en que cambia el estado del objeto, como resultado de los eventos que llegan a él. Los diagramas de estados se dibujan para una sola clase, mostrando el comportamiento de un solo objeto durante todo su ciclo de vida.

ELEMENTOS DE LOS DIAGRAMAS DE ESTADO

- **Estado Inicial:** círculo negro.
- **Estado Final:** círculo concéntrico (Doble Circulo).
- **Estado:** rectángulo de bordes redondeados, identifica un periodo de tiempo del objeto (no instantáneo) en el cual el objeto está esperando alguna operación, tiene cierto estado característico o puede recibir cierto tipo de estímulos.
- **Transacción.** Flecha Unidireccional.

3.3.1 DIAGRAMA DE ESTADOS SGMAS

Para tener una mayor comprensión de los siguientes diagramas de estado procedemos a realizar un cuadro de interpretación de los mismos:

CUADRO DE INTERPRETACION DE DIAGRAMAS DE ESTADO

DIAGRAMA	INTERPRETACIÓN
3.3.1.1	INGRESO AL SISTEMA: Proceso mediante el cual el Sistema verifica los datos ingresados por el usuario y los autentifica para el despliegue de las interfaces dependiendo del perfil que este posea.
3.3.1.2	INGRESAR FAMILIA: Proceso mediante el cual se ingresa al Sistema una nueva familia dependiendo del producto a ser ingresado (insumos médicos o medicamentos).
3.3.1.3	MODIFICAR FAMILIA: Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.
3.3.1.4	ELIMINAR FAMILIA: Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.
3.3.1.5	INGRESAR BODEGA: Proceso mediante el cual se ingresa al Sistema una nueva bodega para almacenamiento de los medicamentos a ser ingresados.
3.3.1.6	MODIFICAR BODEGA: Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.
3.3.1.7	ELIMINAR BODEGA: Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.
3.3.1.8	INGRESAR FÓRMULA FARMACÉUTICA:

Proceso mediante el cual se ingresa al Sistema una nueva fórmula farmacéutica dependiendo de las características de los medicamentos a ser ingresados.

MODIFICAR FÓRMULA FARMACÉUTICA:

3.3.1.9 Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

ELIMINAR FÓRMULA FARMACÉUTICA:

3.3.1.10 Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

INGRESAR RESPONSABLE:

3.3.1.11 Proceso mediante el cual se ingresa al Sistema un responsable del manejo de una de las bodegas ya existentes en la aplicación.

MODIFICAR RESPONSABLE:

3.3.1.12 Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

ELIMINAR RESPONSABLE:

3.3.1.13 Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

INGRESAR CONCENTRACIÓN:

3.3.1.14 Proceso mediante el cual se ingresa al Sistema una nueva concentración dependiendo de las características de los medicamentos a ser ingresados.

MODIFICAR CONCENTRACIÓN:

3.3.1.15 Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

ELIMINAR CONCENTRACIÓN:

3.3.1.16 Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del

usuario para ser guardado en el sistema.

INGRESAR PROVEEDOR:

3.3.1.17 Proceso mediante el cual se ingresa al Sistema los datos generales del proveedor para la adquisición de medicamentos.

MODIFICAR PROVEEDOR:

3.3.1.18 Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

ELIMINAR PROVEEDOR:

3.3.1.19 Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

INGRESAR TIPO DOCUMENTO CONTABLE:

3.3.1.20 Proceso mediante el cual se ingresa al Sistema un tipo de documento contable según como se ingresen los pedidos (nota de venta, factura, recibo, etc.).

MODIFICAR TIPO DOCUMENTO CONTABLE:

3.3.1.21 Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

ELIMINAR TIPO DOCUMENTO CONTABLE:

3.3.1.22 Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

INGRESAR REGISTRO DOCUMENTO CONTABLE:

3.3.1.23 Proceso mediante el cual se ingresa al Sistema una nueva fórmula farmacéutica dependiendo de las características de los medicamentos a ser ingresados.

MODIFICAR REGISTRO DOCUMENTO CONTABLE:

3.3.1.24 Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

3.3.1.25	ELIMINAR REGISTRO DOCUMENTO CONTABLE: Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.
3.3.1.26	INGRESAR UBICACIÓN FÍSICA: Proceso mediante el cual se asigna en el Sistema una ubicación física a cada medicamento ingresado para mantener un orden en el almacenamiento de los medicamentos.
3.3.1.27	INGRESAR BAJAS DE PRODUCTOS: Proceso mediante el cual se ingresan medicamentos dados de baja dependiendo del siniestro que lo ha provocado.
3.3.1.28	GENERAR REPORTES: Proceso que muestra un listado completo de los medicamentos que se encuentran ingresados en el sistema con los datos más relevantes del mismo con filtros de búsqueda incremental para su posterior impresión.
3.3.1.29	INGRESAR NUEVO MEDICAMENTO: Proceso mediante el cual se ingresa al Sistema los datos generales de cada uno de los medicamentos para su posterior utilización en el ingreso de lotes de medicación.
3.3.1.30	MODIFICAR MEDICAMENTO: Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.
3.3.1.31	ELIMINAR MEDICAMENTO: Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.
3.3.1.32	INGRESAR LOTE DE PRODUCTOS: Ingresados los datos generales del medicamento se

	realiza el proceso de ingreso de un nuevo lote de medicación con los datos relevantes para llevar un inventario de los productos ingresados.
3.3.1.33	MODIFICAR LOTE DE PRODUCTOS: Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.
3.3.1.34	ELIMINAR LOTE DE PRODUCTOS: Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.
3.3.1.35	REALIZAR DESPACHOS DE PRODUCTOS: Se procede a realizar una cabecera con los datos generales del despacho para posteriormente elegir cada uno de los medicamentos a despacharse con su respectivo documento referencial previamente aprobado para la verificación de los productos despachados e impresión del comprobante.
3.3.1.36	MODIFICAR DESPACHOS DE PRODUCTOS: Proceso mediante el cual se modifican los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.
3.3.1.37	ELIMINAR DESPACHOS DE PRODUCTOS: Proceso mediante el cual se eliminan los datos ya ingresados, dependiendo de las necesidades del usuario para ser guardado en el sistema.

Tabla 3.3.1 Interpretación Diagrama de Estados SGMAS[27]

[27] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.1 Diagrama De Estado Ingresar Al Sistema:

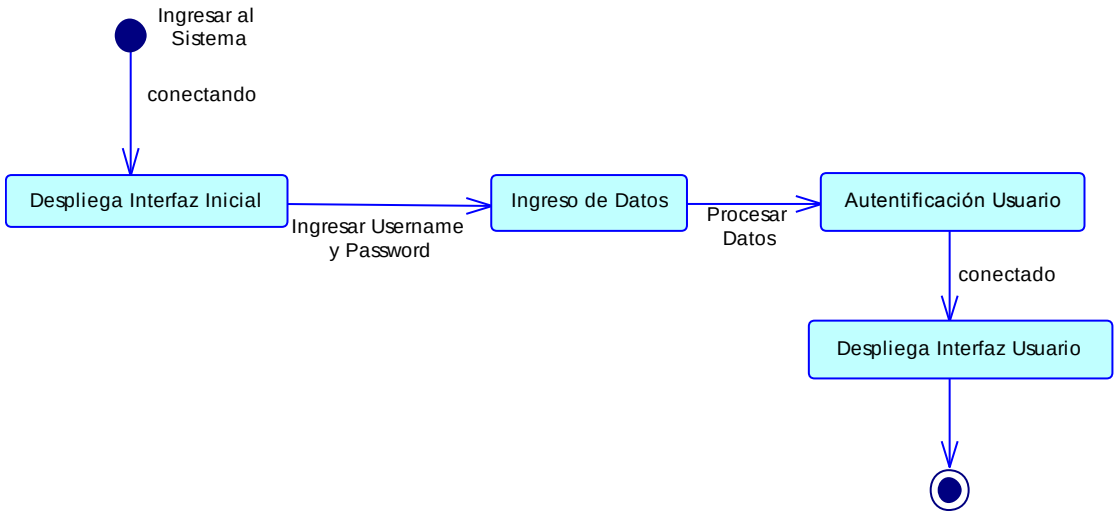


Figura. 3.3.1.1. Diagrama de Estado Ingresar al Sistema[28]

[28] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.2 Diagrama De Estado Ingresar Familia

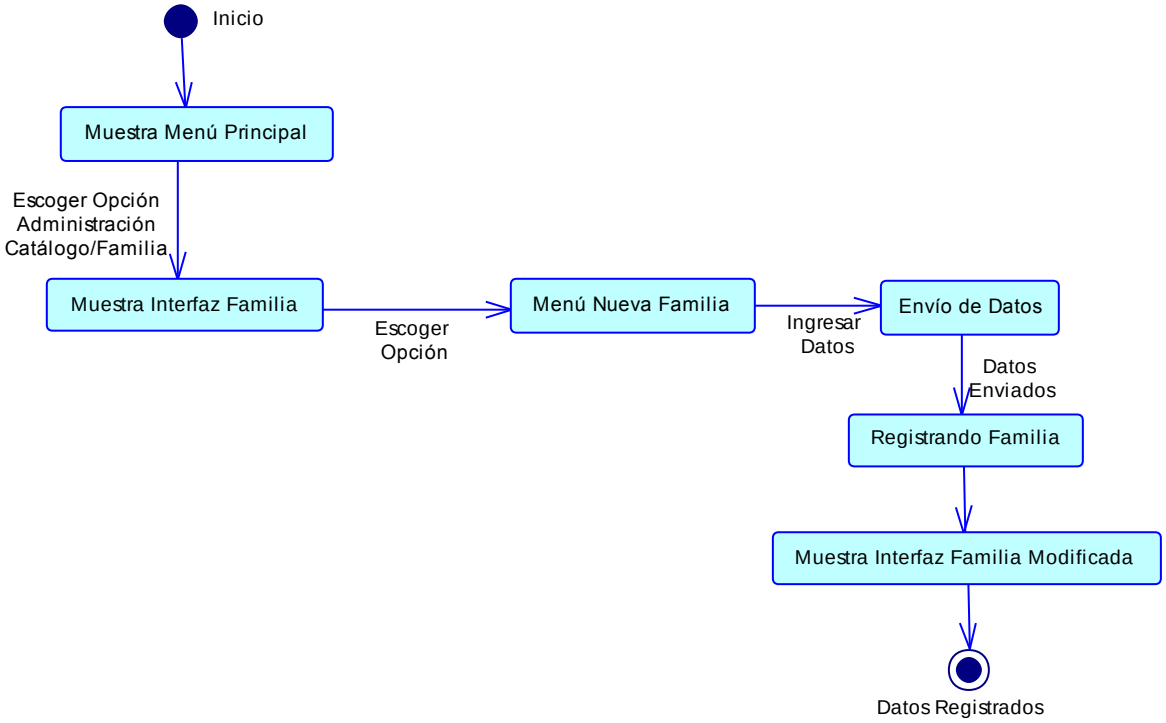


Figura. 3.3.1.2. Diagrama de Estado Ingresar Familia[29]

[29] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.3 Diagrama De Estado Modificar Familia

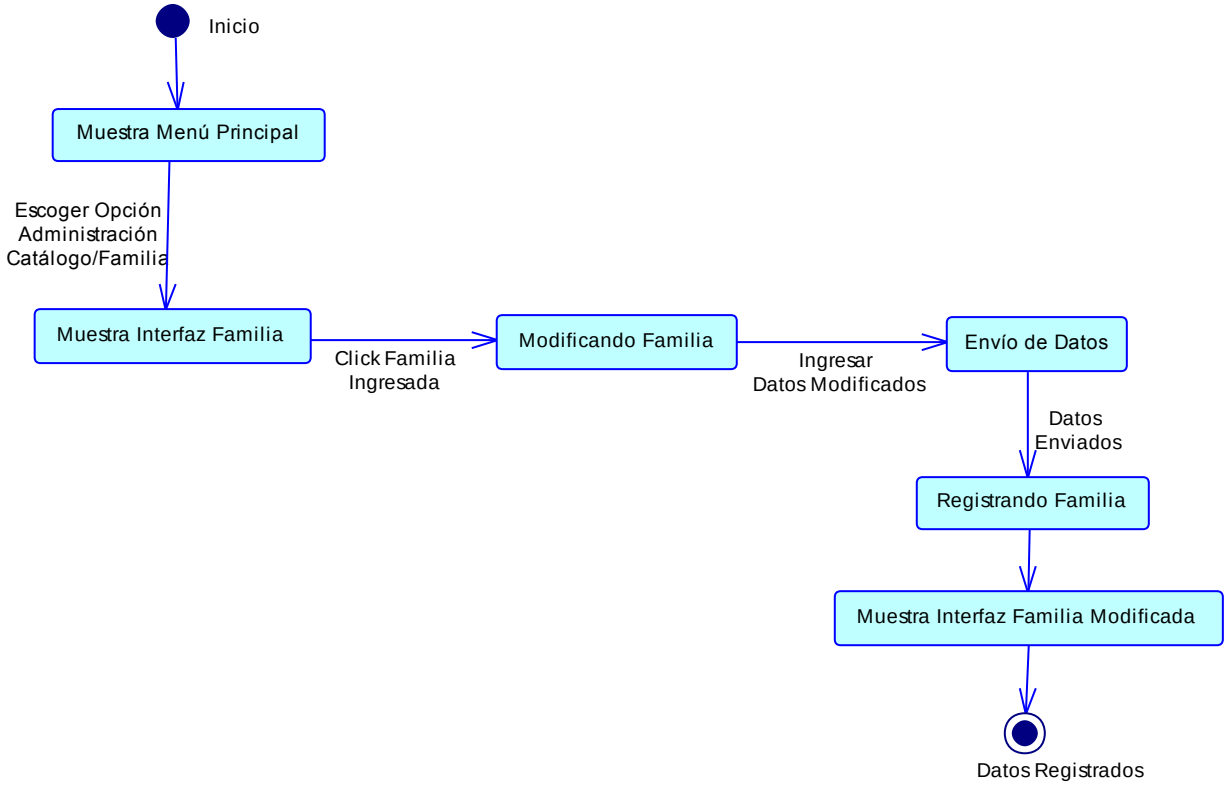


Figura. 3.3.1.3. Diagrama de Estado Modificar Familia[30]

[30] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.4 Diagrama De Estado Eliminar Familia

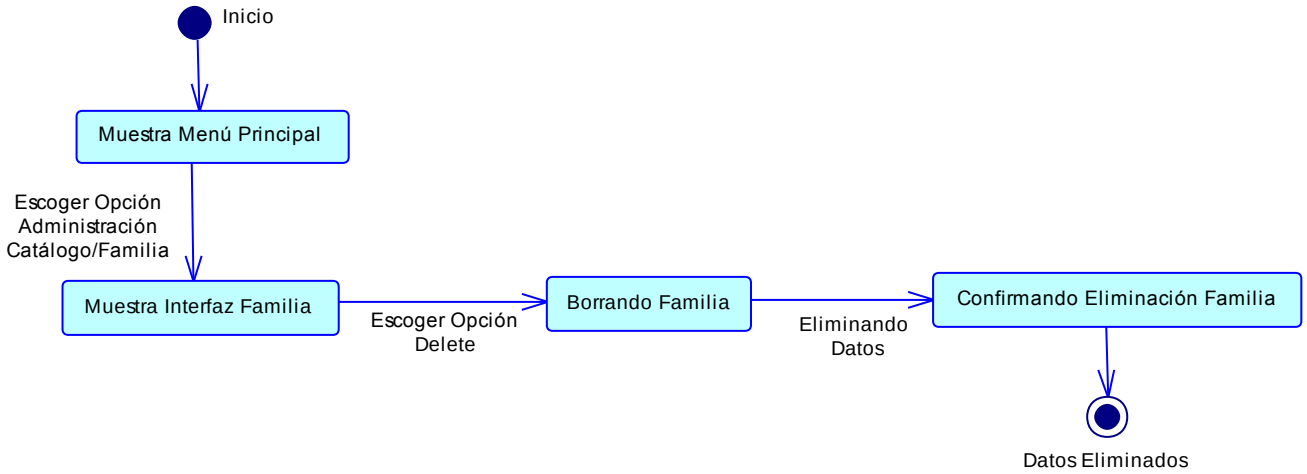


Figura. 3.3.1.4. Diagrama de Estado Eliminar Familia[31]

[31] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.5 Diagrama De Estado Ingresar Bodega

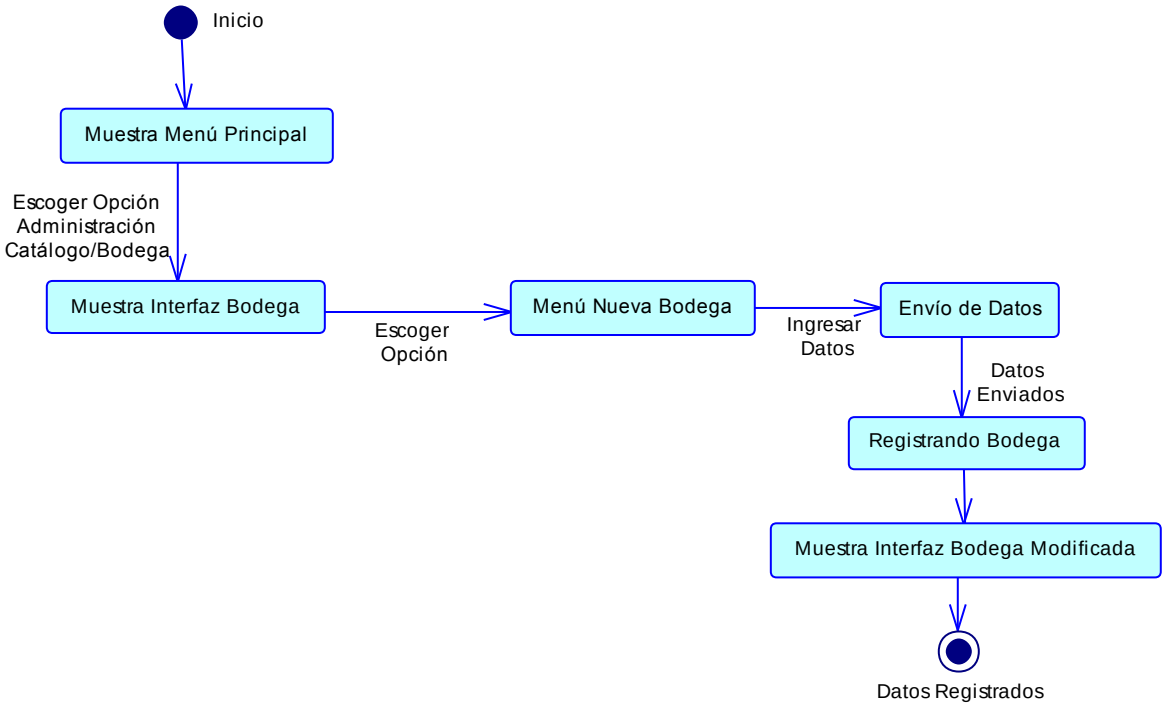


Figura. 3.3.1.5. Diagrama de Estado Ingresar Bodega[32]

[32] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.6 Diagrama De Estado Modificar Bodega

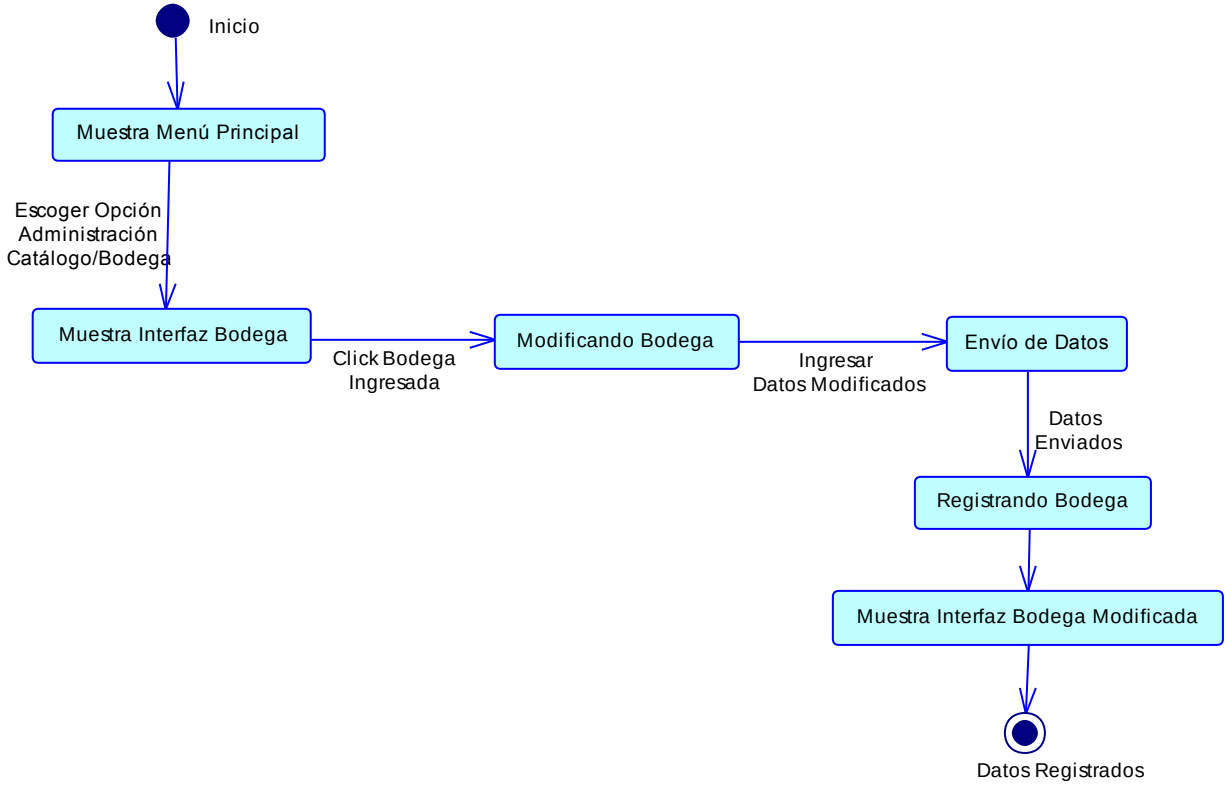


Figura. 3.3.1.6. Diagrama de Estado Modificar Bodega[33]

[33] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.7 Diagrama De Estado Eliminar Bodega

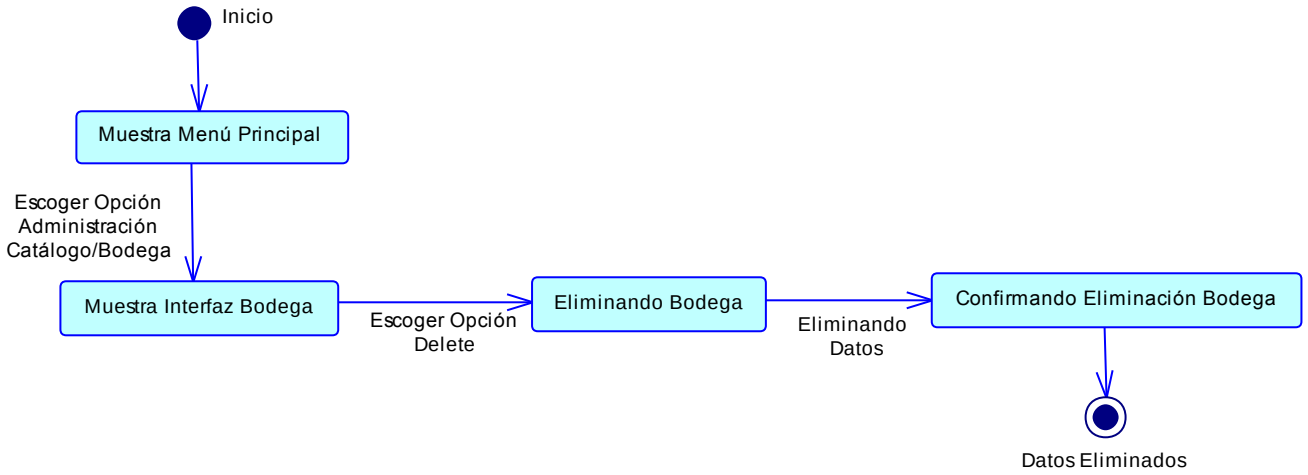


Figura. 3.3.1.7. Diagrama de Estado Eliminar Bodega[34]

[34] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.8 Diagrama De Estado Ingresar Fórmula Farmacéutica

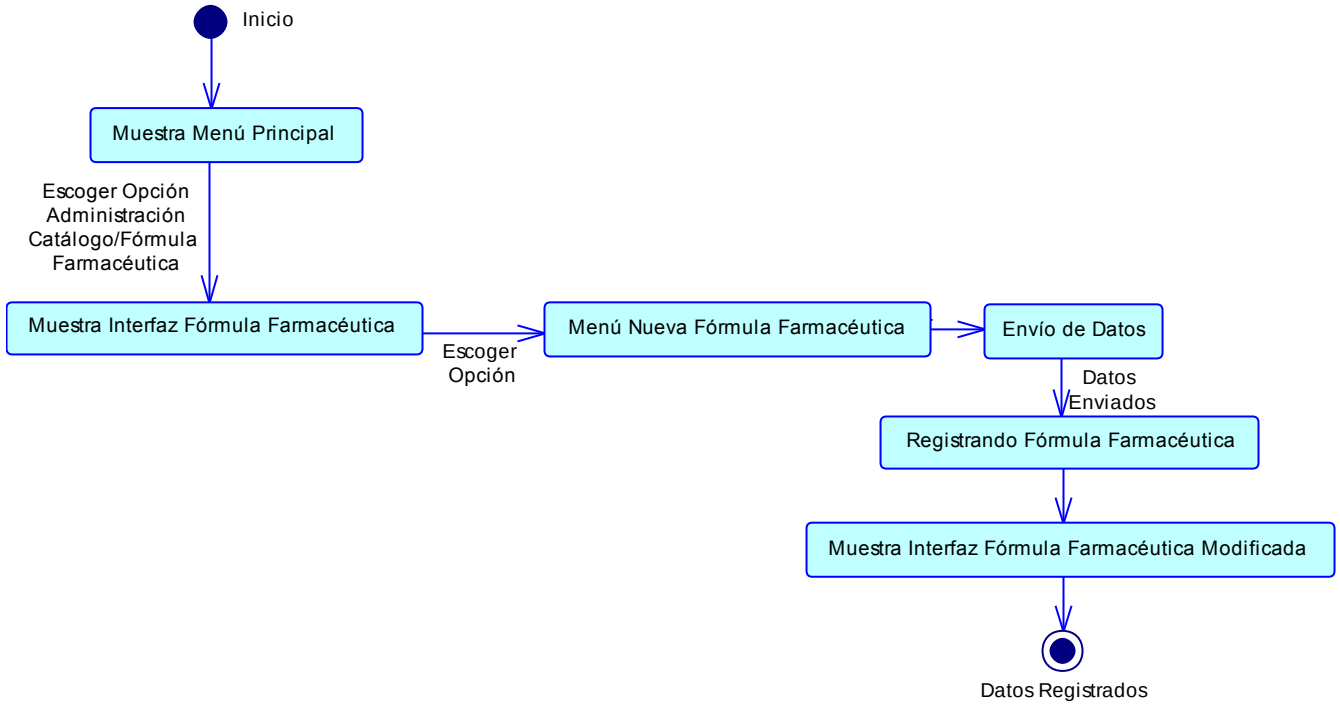


Figura. 3.3.1.8. Diagrama de Estado Ingresar Fórmula Farmacéutica[35]

[35] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.9 Diagrama De Estado Modificar Fórmula Farmacéutica

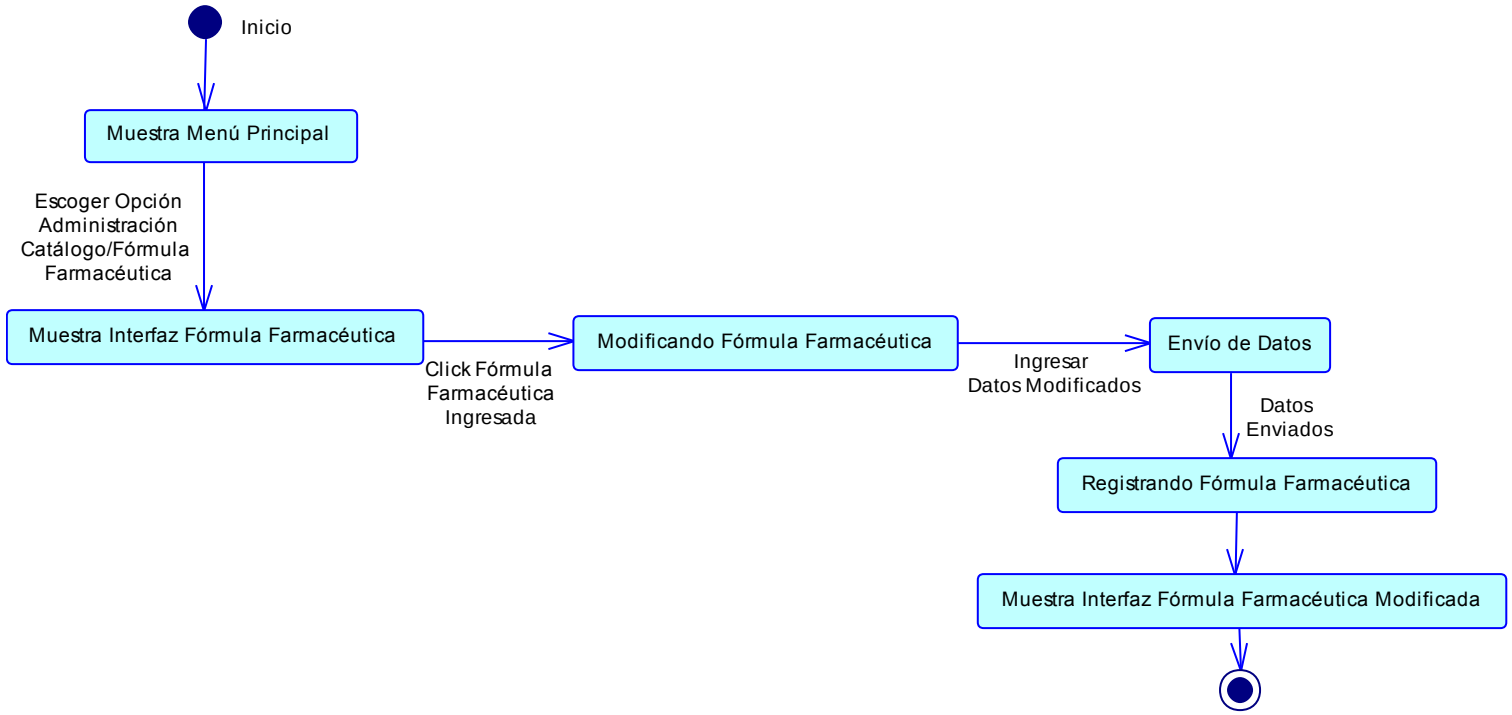


Figura. 3.3.1.9. Diagrama de Estado Modificar Fórmula Farmacéutica[36]

[36] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.10 Diagrama De Estado Eliminar Fórmula Farmacéutica

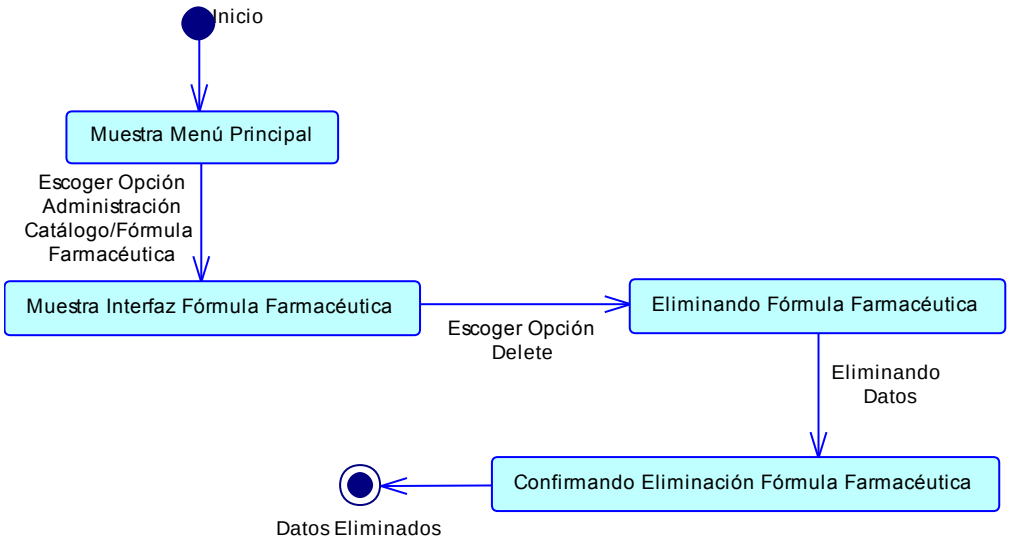


Figura. 3.3.1.10. Diagrama de Estado Eliminar Fórmula Farmacéutica[37]

[37] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.11 Diagrama De Estado Ingresar Responsable

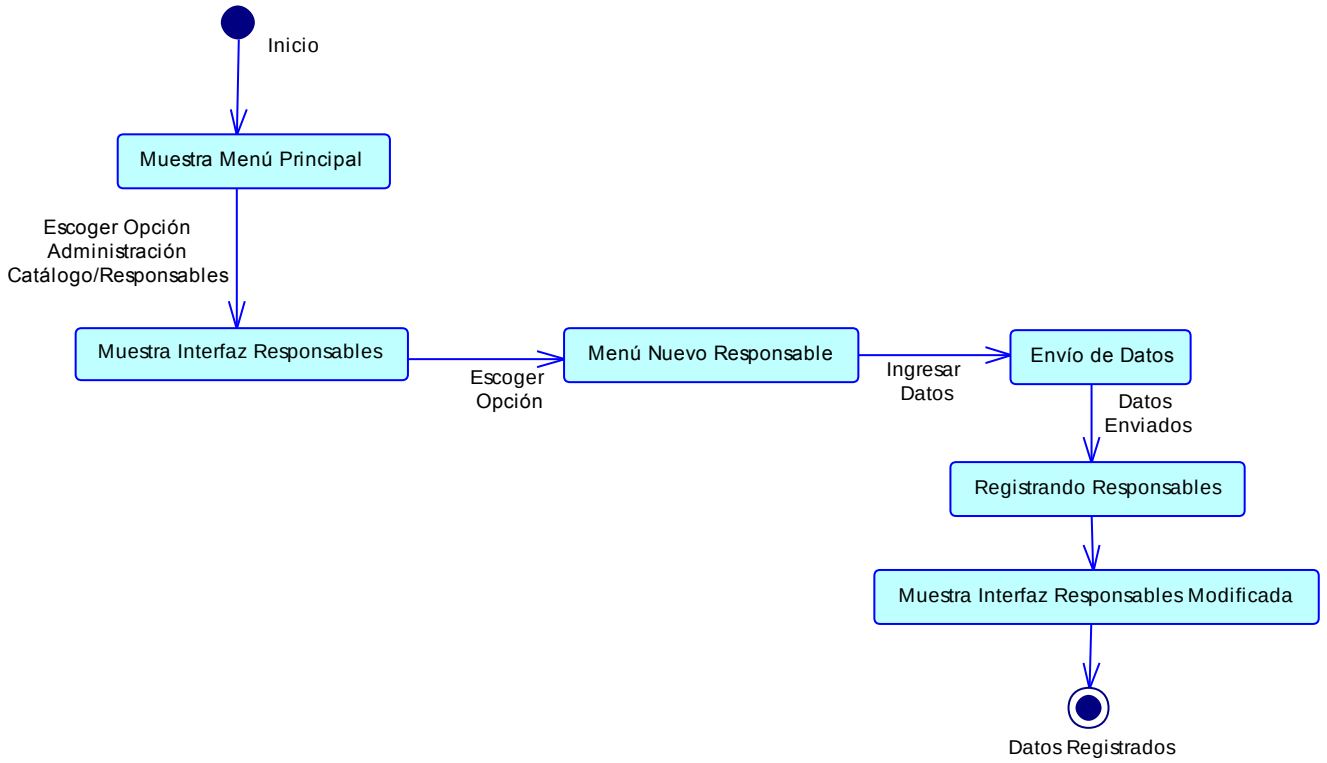


Figura. 3.3.1.11. Diagrama de Estado Ingresar Responsable[38]

[38] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.12 Diagrama De Estado Modificar Responsable

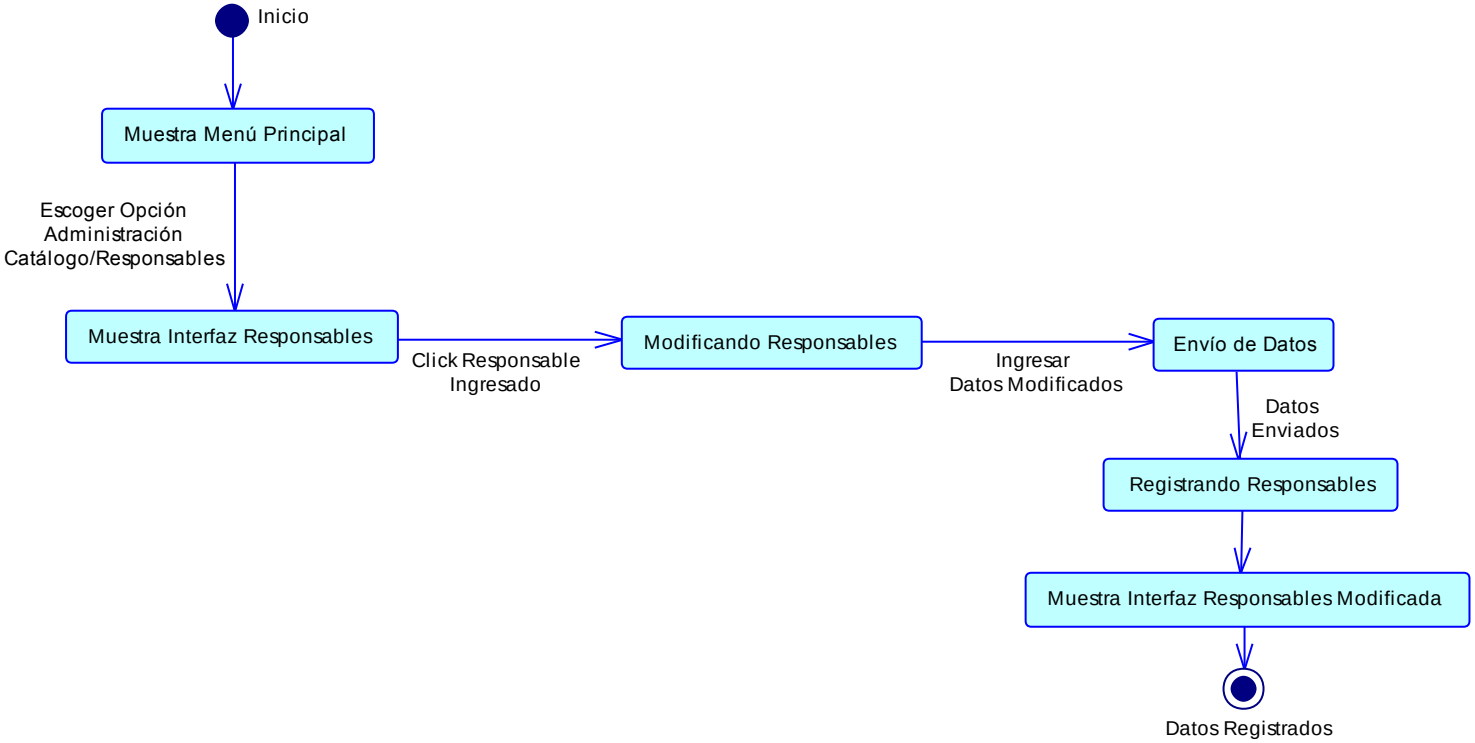


Figura. 3.3.1.12. Diagrama de Estado Modificar Responsable[39]

[39] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.13 Diagrama De Estado Eliminar Responsable

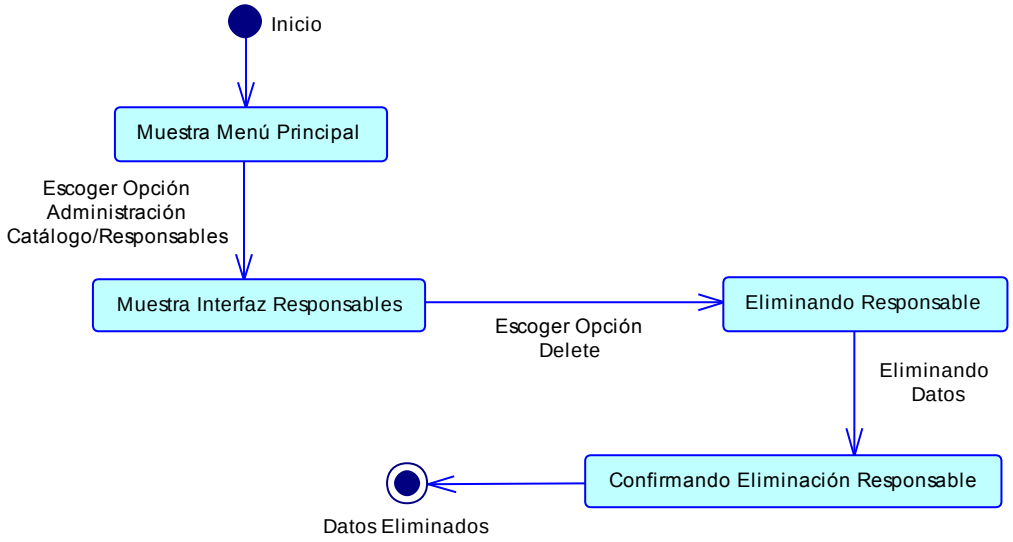


Figura. 3.3.1.13. Diagrama de Estado Eliminar Responsable[40]

[40] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.14 Diagrama De Estado Ingresar Concentración

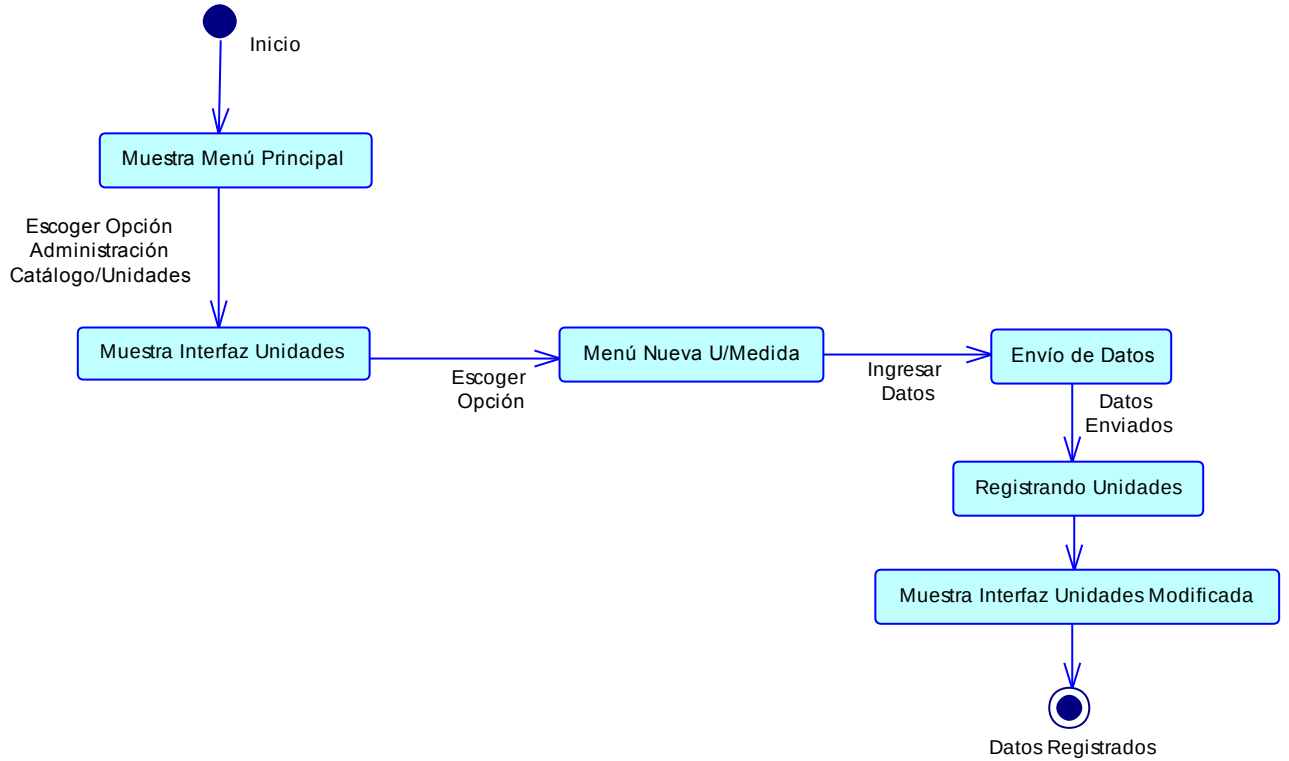


Figura. 3.3.1.14. Diagrama de Estado Ingresar Concentración[41]

[41] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.15 Diagrama De Estado Modificar Concentración

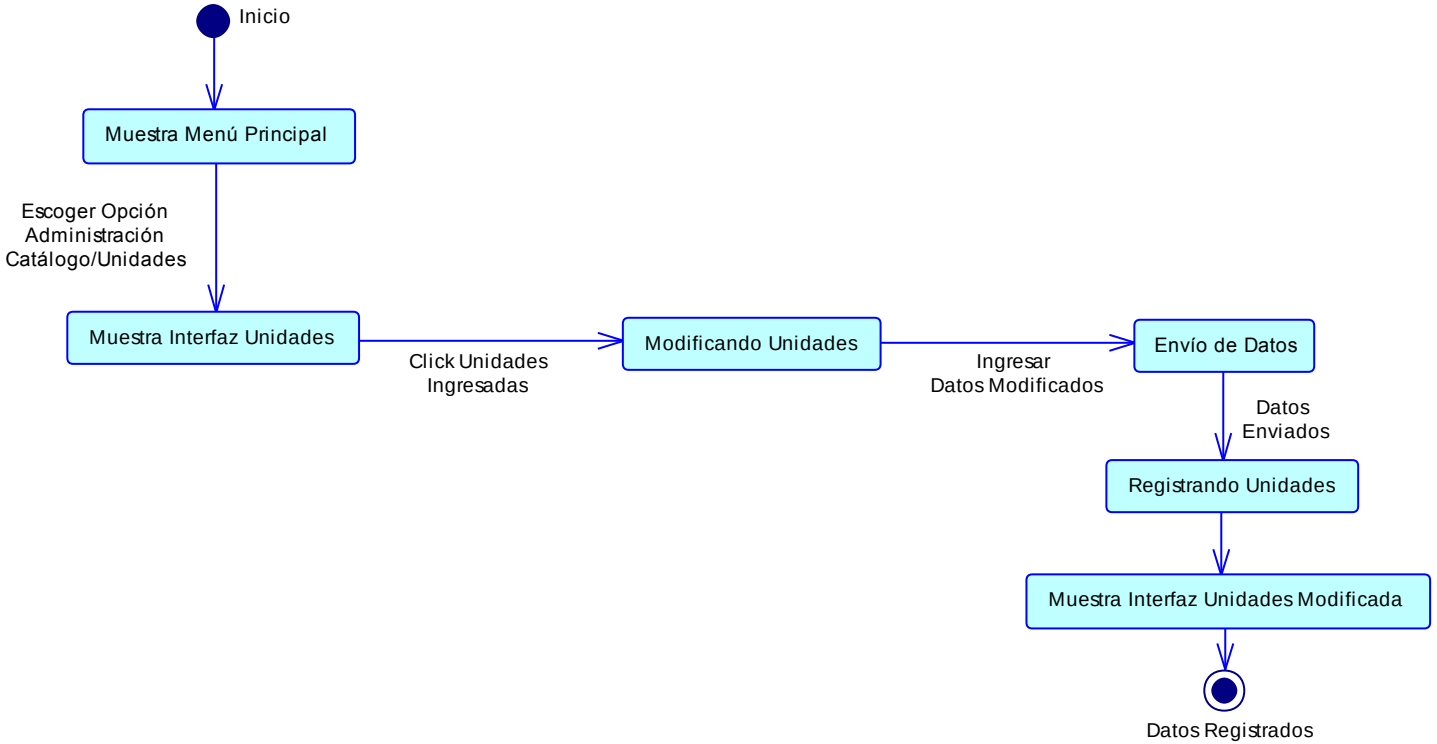


Figura. 3.3.1.15. Diagrama de Estado Modificar Concentración[42]

[42] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.16 Diagrama De Estado Eliminar Concentración

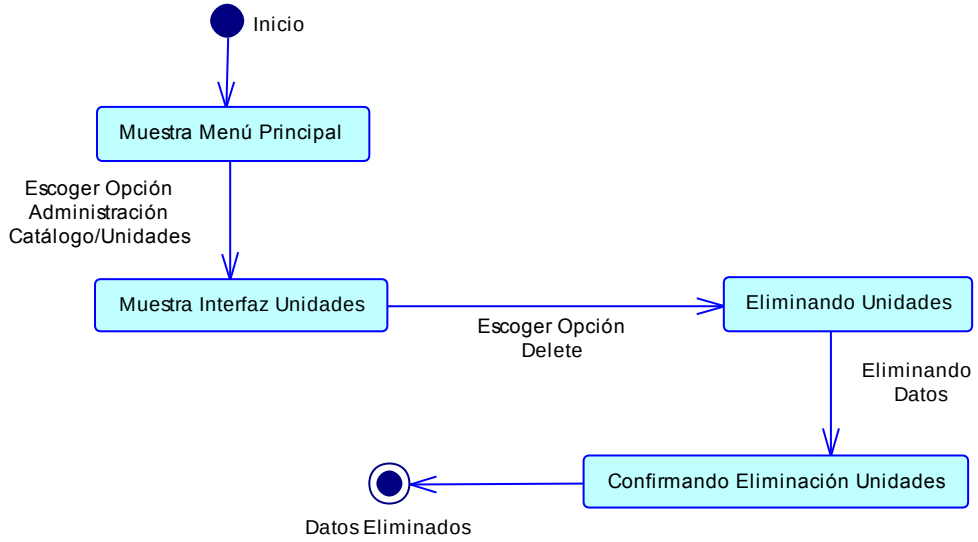


Figura. 3.3.1.16. Diagrama de Estado Eliminar Concentración[43]

[43] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.17 Diagrama De Estado Ingresar Proveedor

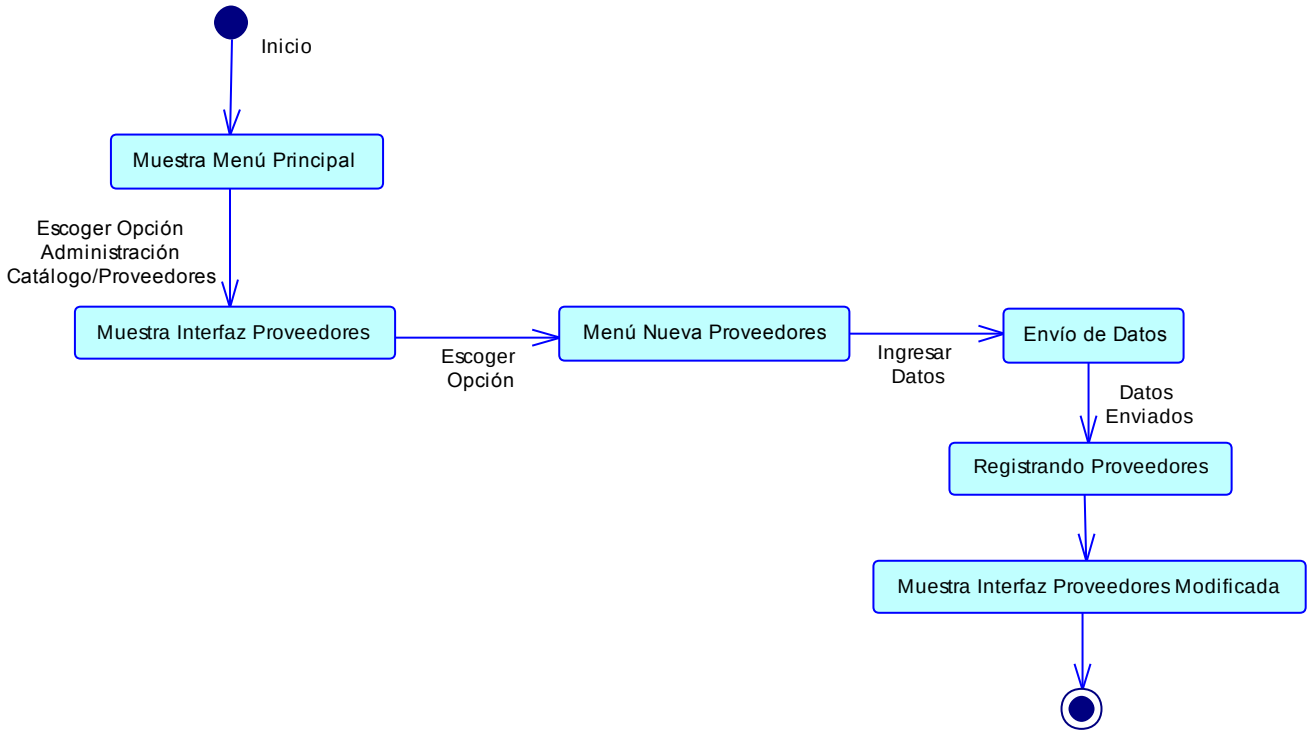


Figura. 3.3.1.17. Diagrama de Estado Ingresar Proveedor[44]

[44] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.18 Diagrama De Estado Modificar Proveedor

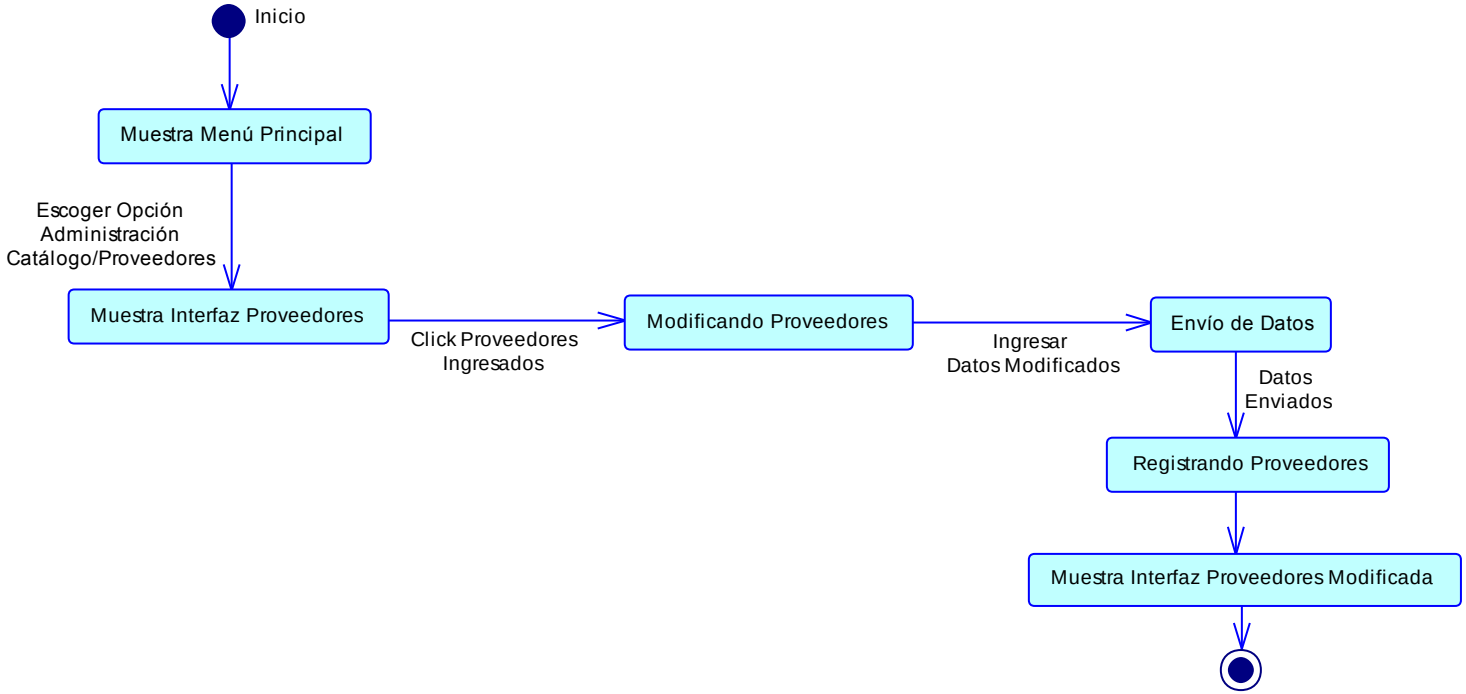


Figura. 3.3.1.18. Diagrama de Estado Modificar Proveedor[45]

[45] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.19 Diagrama De Estado Eliminar Proveedor

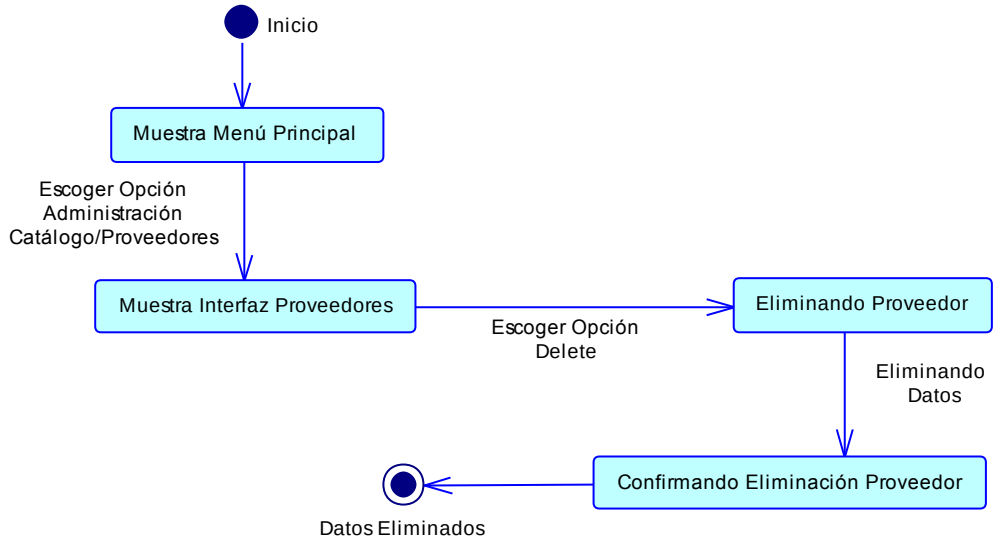


Figura. 3.3.1.19. Diagrama de Estado Eliminar Proveedor[46]

[46] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.20 Diagrama De Estado Ingreso Tipo Documento Contable

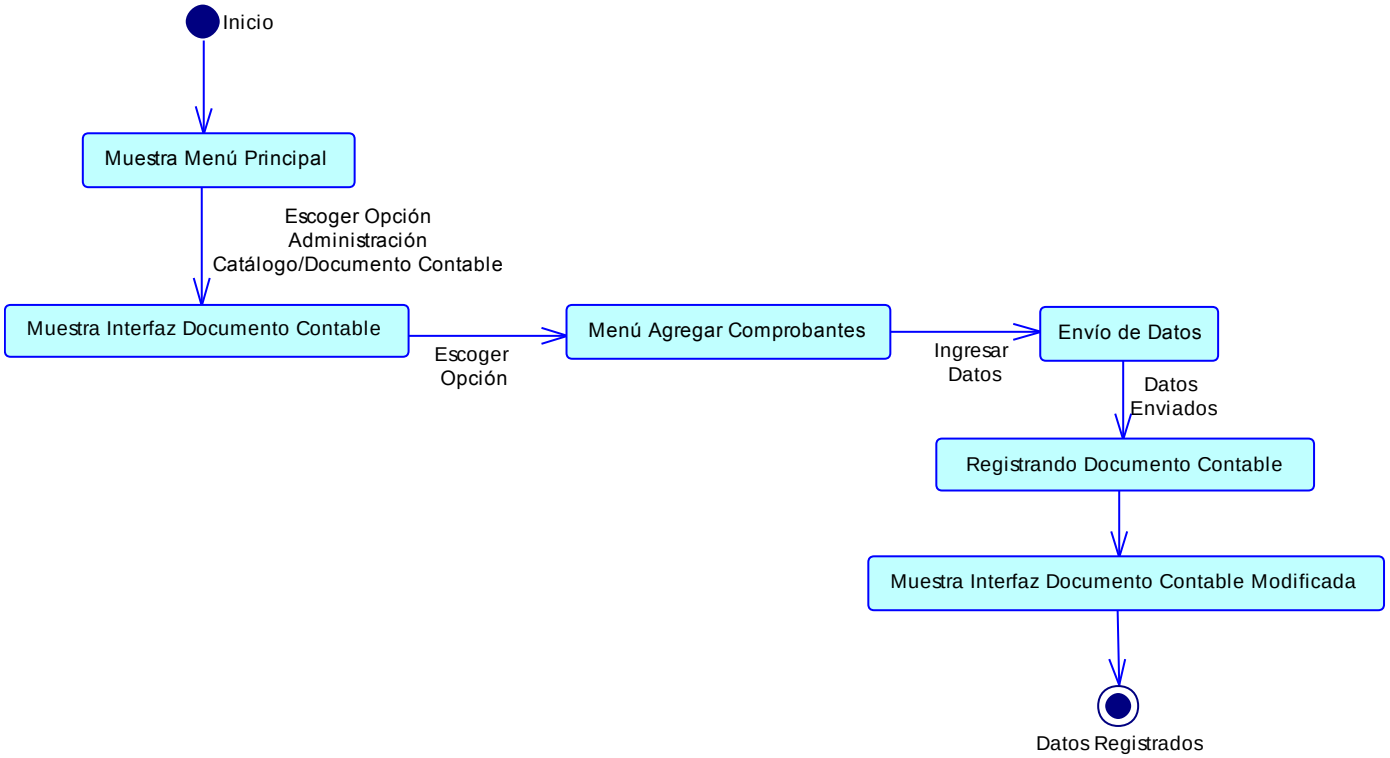


Figura. 3.3.1.20. Diagrama de Estado Ingresar Tipo Documento Contable[47]

[47] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.21 Diagrama De Estado Modificar Tipo Documento Contable

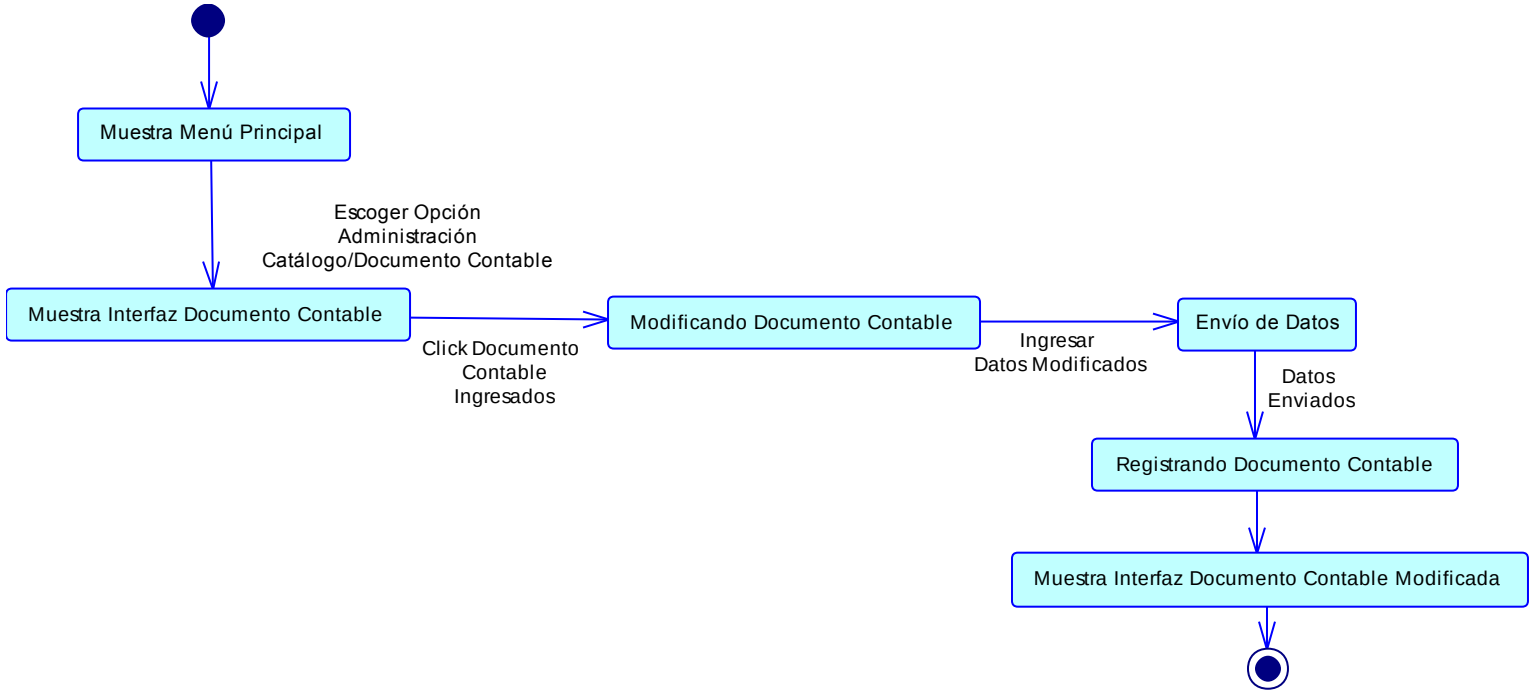


Figura. 3.3.1.21. Diagrama de Estado Modificar Tipo Documento Contable[48]

[48] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.22 Diagrama De Estado Eliminar Tipo Documento Contable

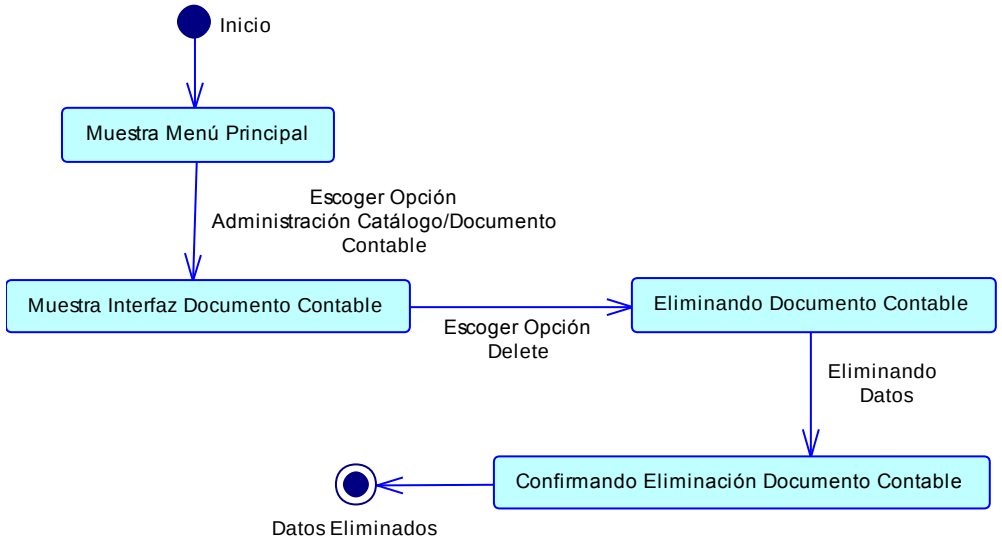


Figura. 3.3.1.22. Diagrama de Estado Eliminar Tipo Documento Contable[49]

[49] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.23 Diagrama De Estado Ingresar Registro Documento Contable

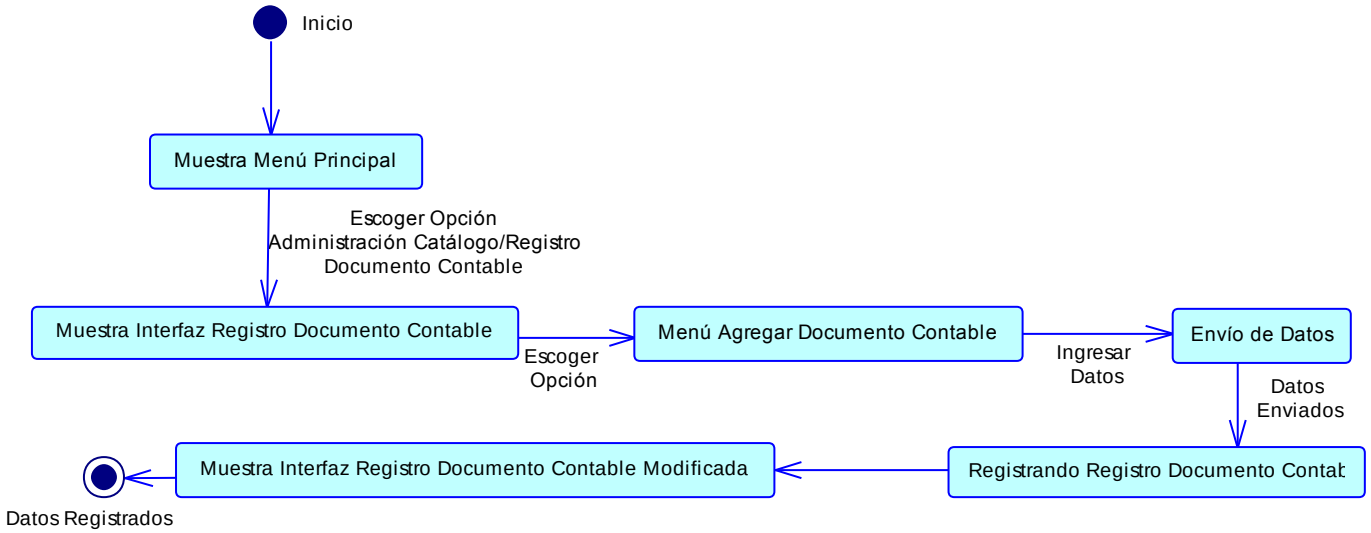


Figura. 3.3.1.23. Diagrama de Estado Ingresar Registro Documento Contable[50]

[50] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.24 Diagrama De Estado Modificar Registro Documento Contable

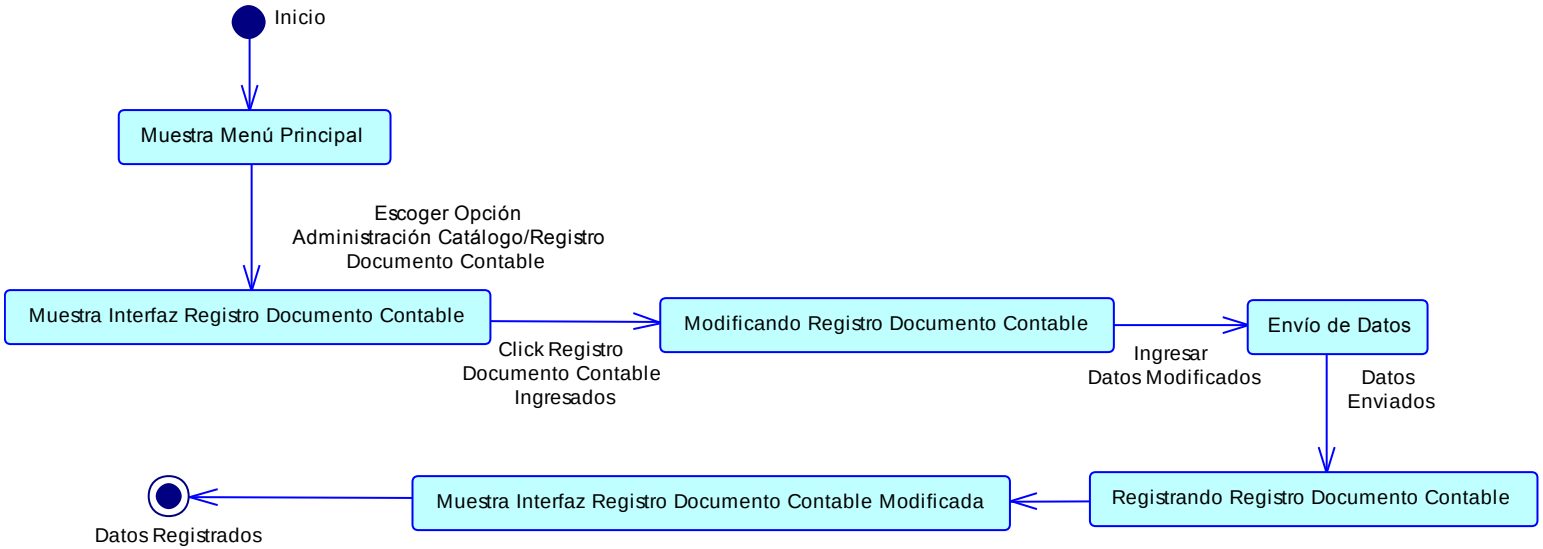


Figura. 3.3.1.24. Diagrama de Estado Modificar Registro Documento Contable[51]

[51] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.25 Diagrama De Estado Eliminar Registro Documento Contable

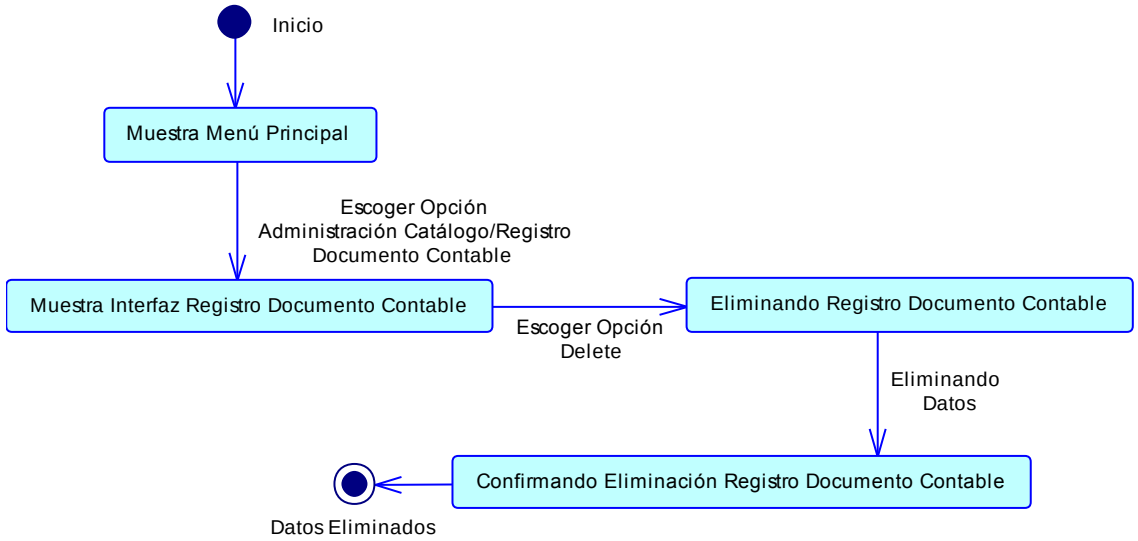


Figura. 3.3.1.25. Diagrama de Estado Eliminar Registro Documento Contable[52]

[52] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.26 Diagrama De Estado Ingresar Ubicación Física

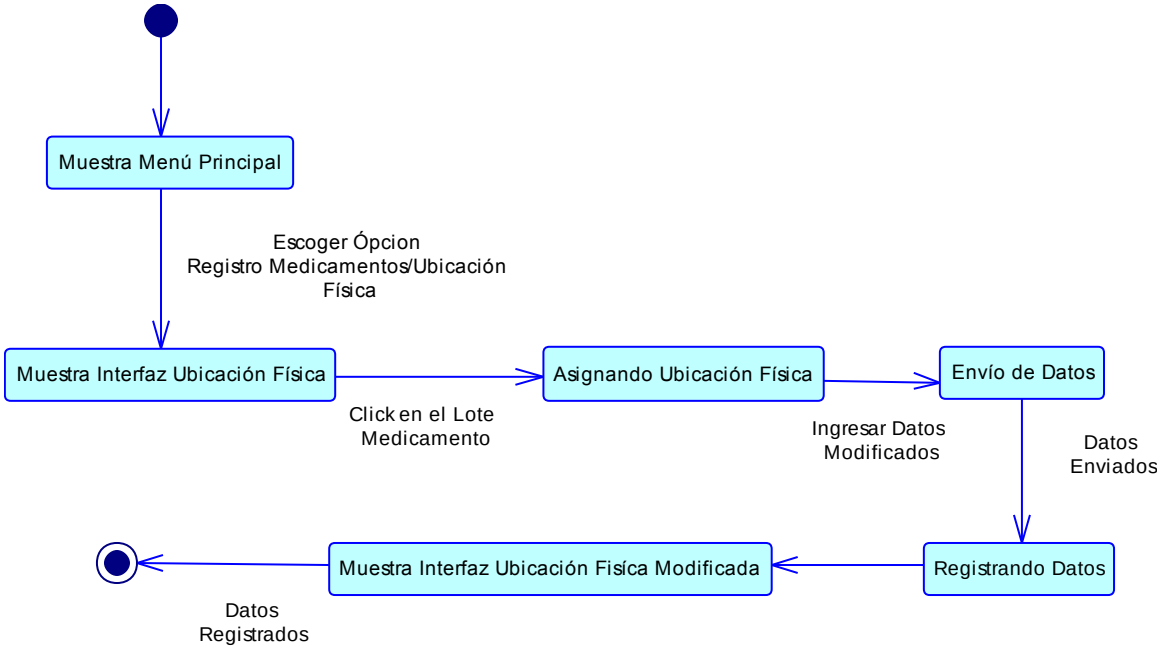


Figura. 3.3.1.26. Diagrama de Estado Ingresar Ubicación Física[53]

[53] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.27 Diagrama De Estado Ingresar Bajas De Productos

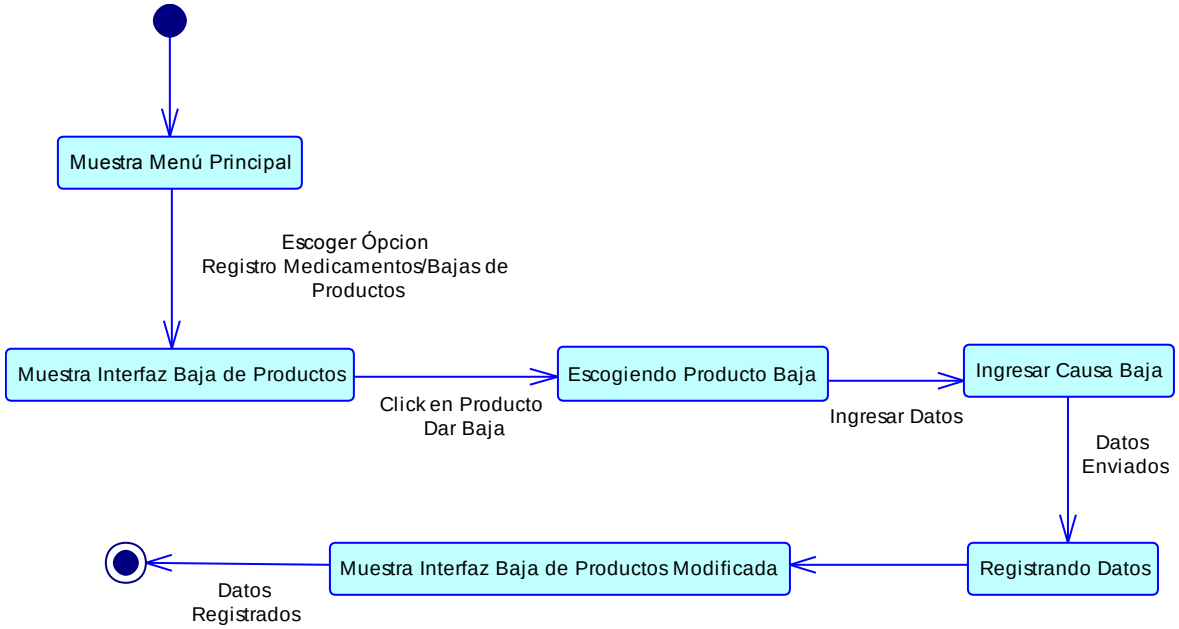


Figura. 3.3.1.27. Diagrama de Estado Ingresar Bajas de Productos[54]

[54] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.28 Diagrama De Estado Generar Reportes

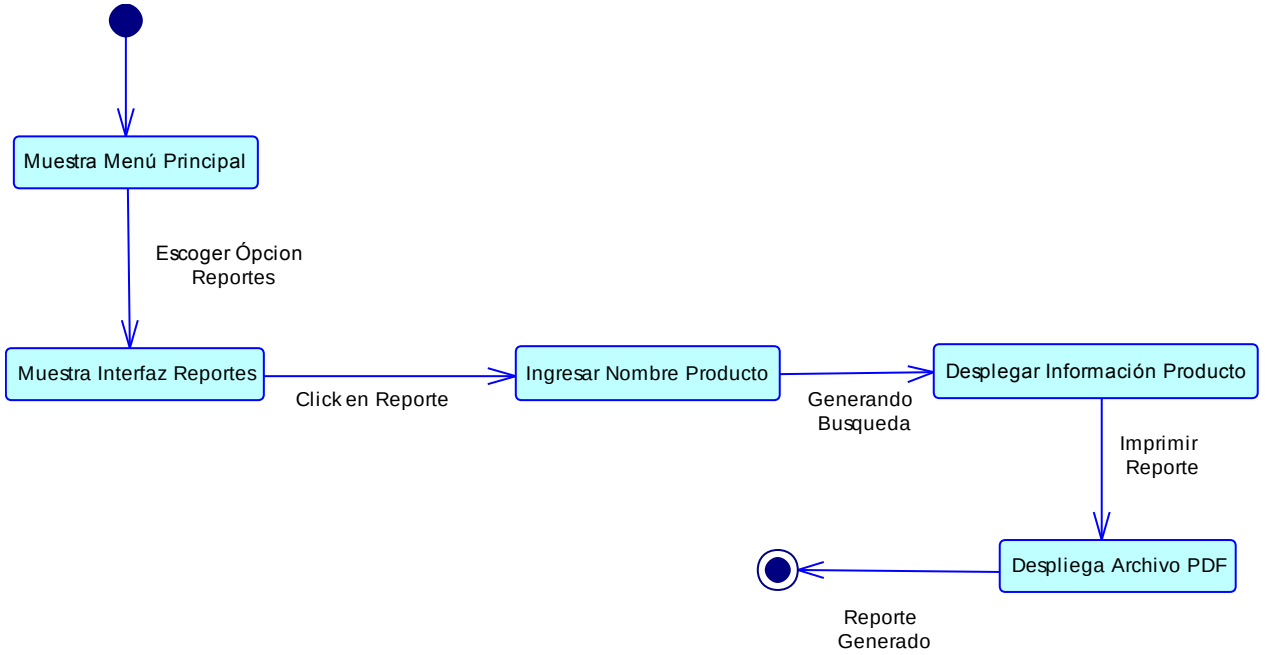


Figura. 3.3.1.28 Diagrama de Estado Reportes por Stock Lotes[55]

[55] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.29 **Diagrama De Estado Ingresar Nuevo Medicamento.**

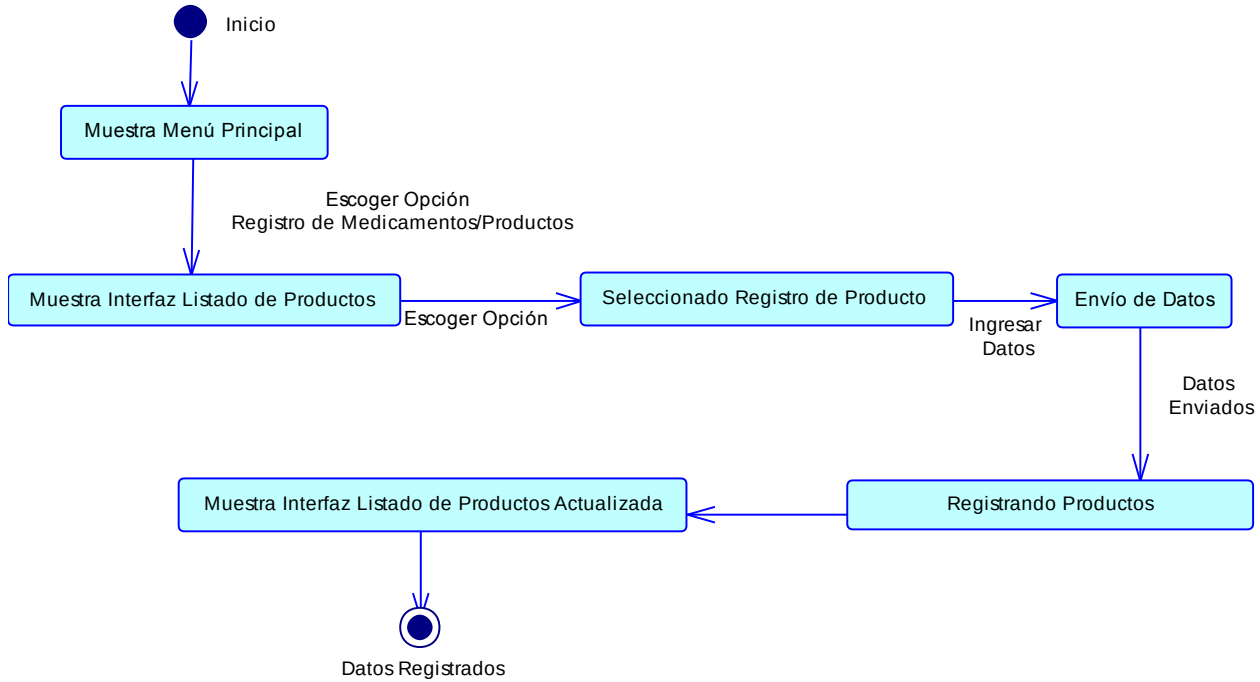


Figura. 3.3.1.29. Diagrama de Estado Ingresar Nuevo Medicamento.[56]

[56] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.30 Diagrama De Estado Modificar Medicamento

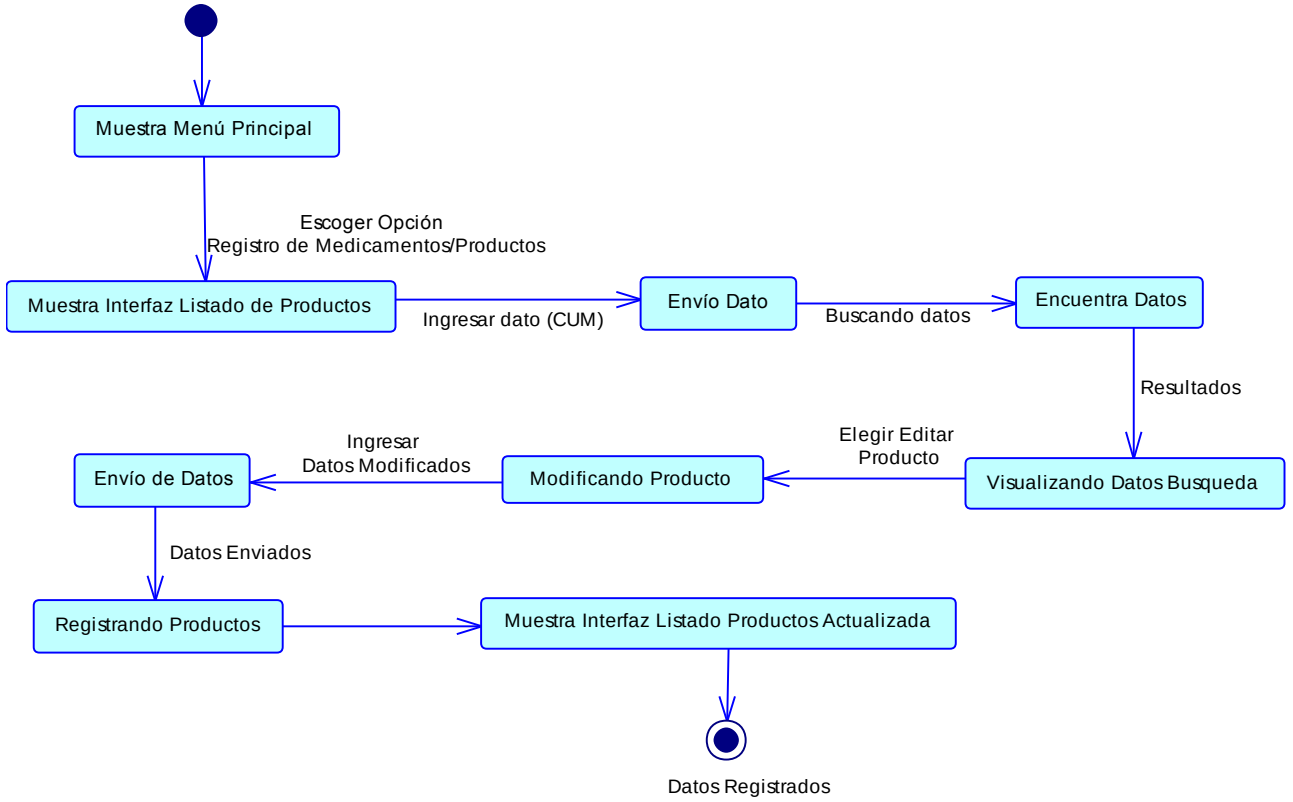


Figura. 3.3.1.30. Diagrama de Estado Modificar Medicamento[57]

[57] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.31 Diagrama De Estado Eliminar Medicamento

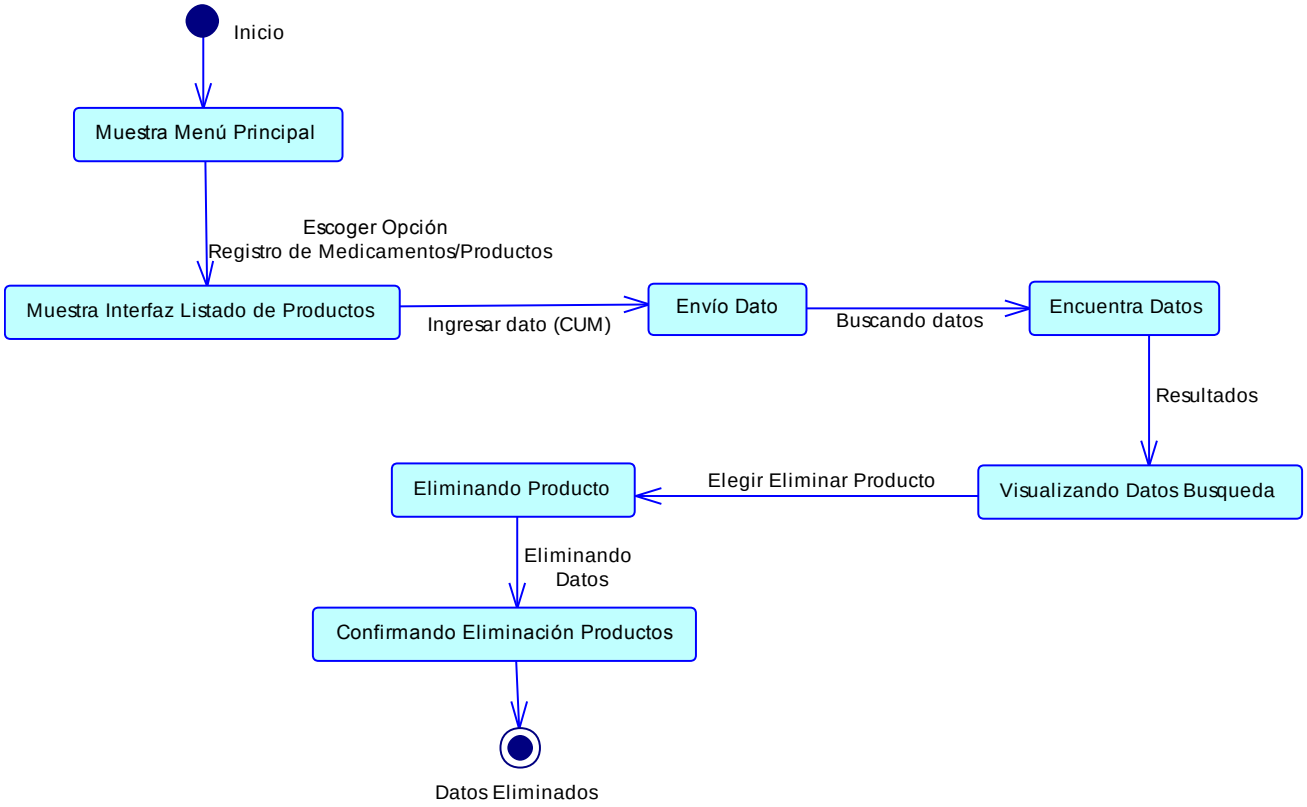


Figura. 3.3.1.31. Diagrama de Estado Eliminar Medicamento[58]

[58] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.32 Diagrama De Estado Ingresar Lote De Productos

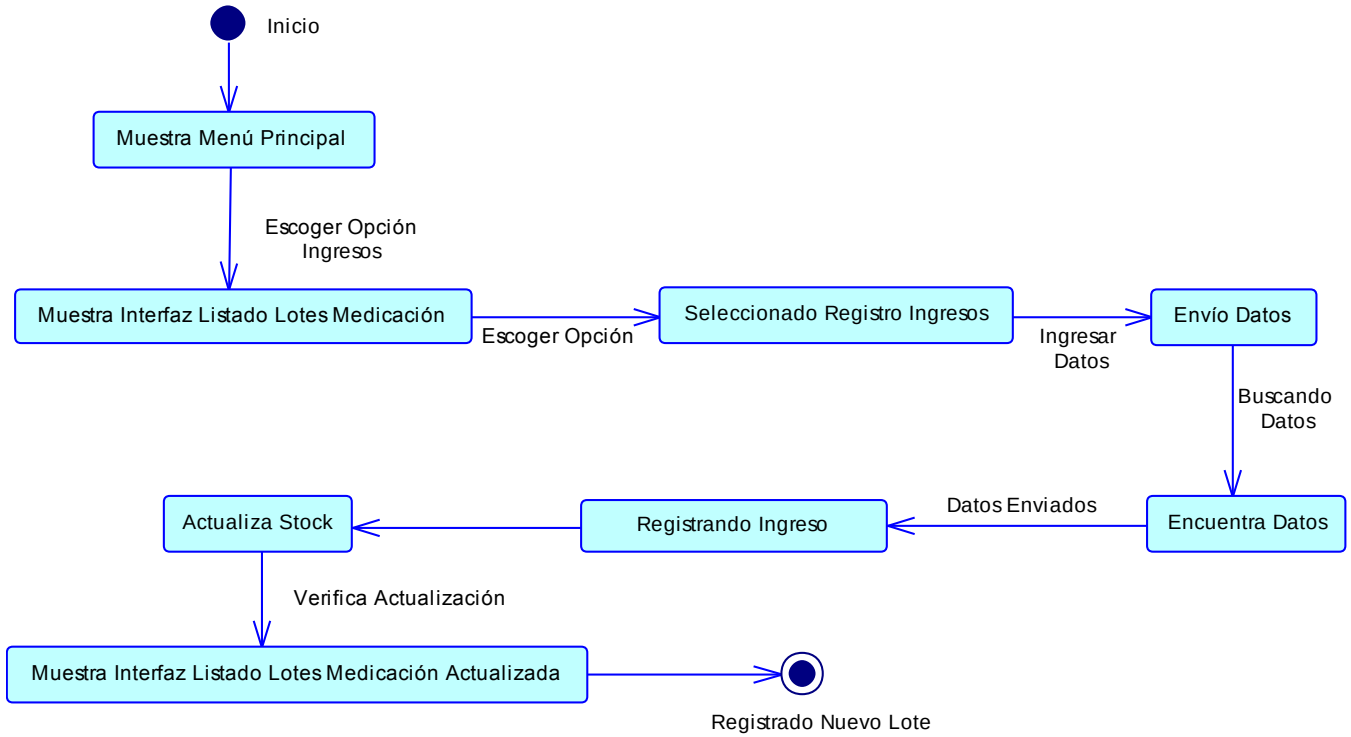


Figura. 3.3.1.32. Diagrama de Estado Ingresar Lote de Productos[59]

[59] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.33 Diagrama De Estado Modificar Lote De Productos

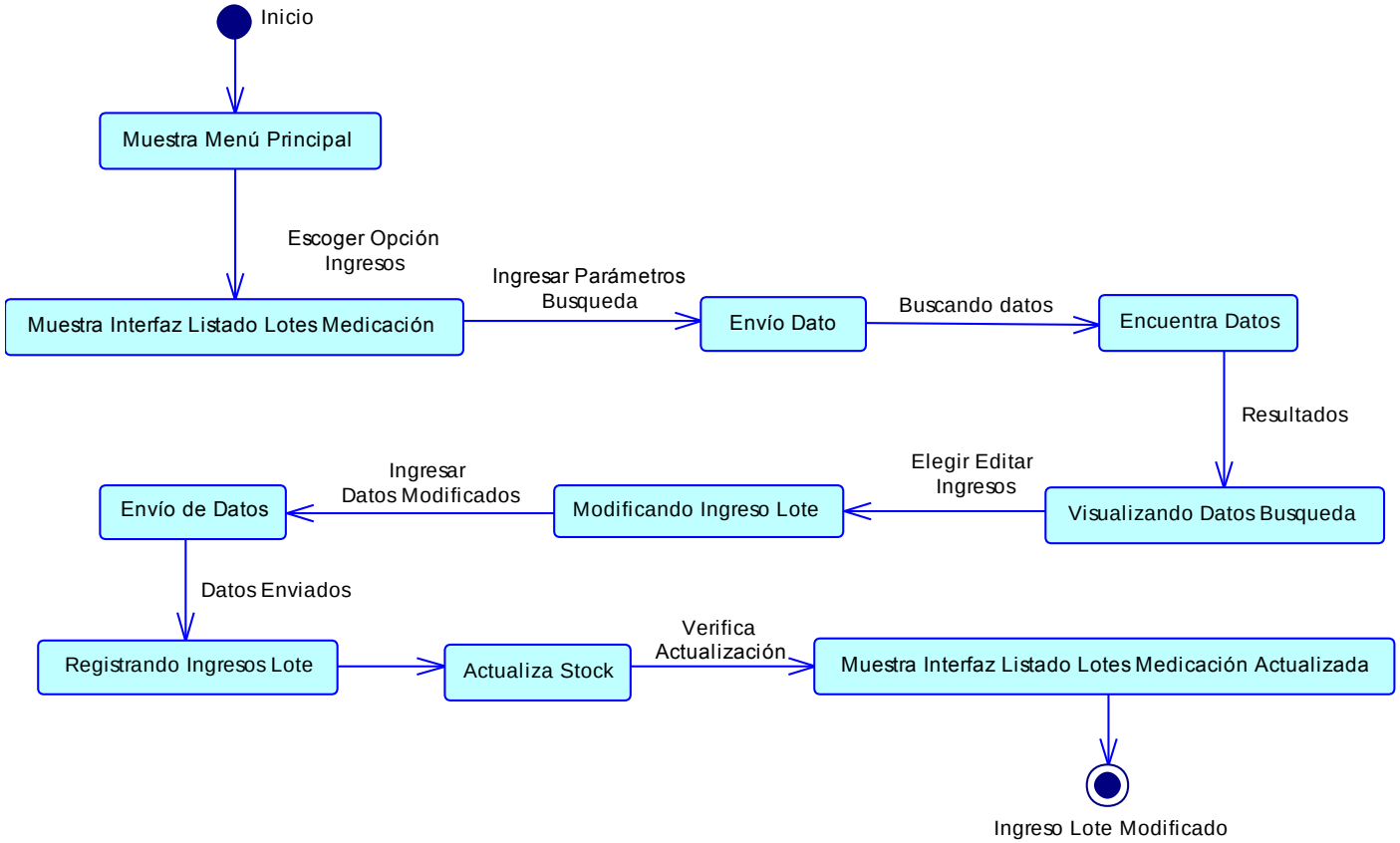


Figura. 3.3.1.33. Diagrama de Estado Modificar Lote de Productos[60]

[60] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.34 Diagrama De Estado Eliminar Lote De Productos

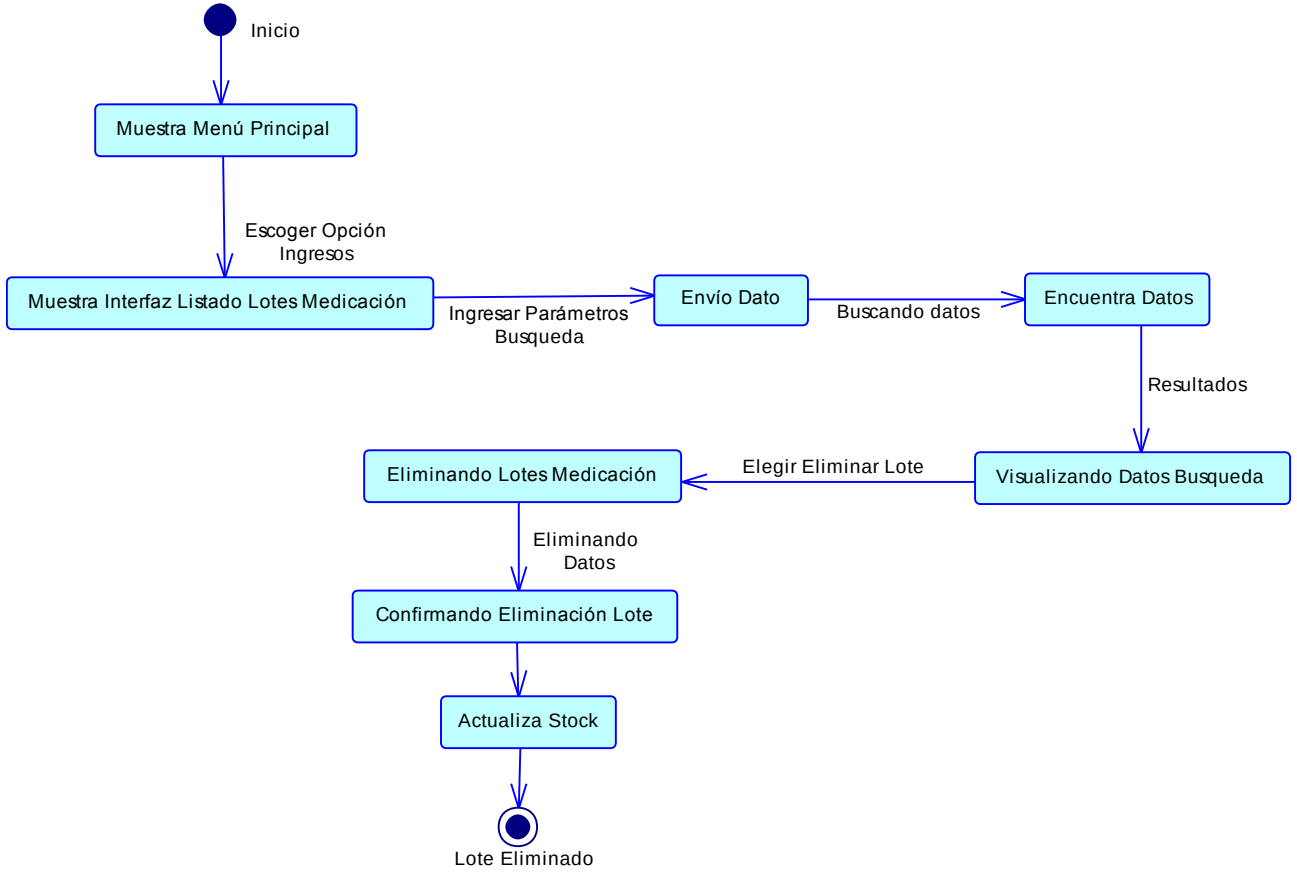


Figura. 3.3.1.34. Diagrama de Estado Eliminar Lote de Productos[61]

[61] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.35 Diagrama De Estado Realizar Despachos De Productos

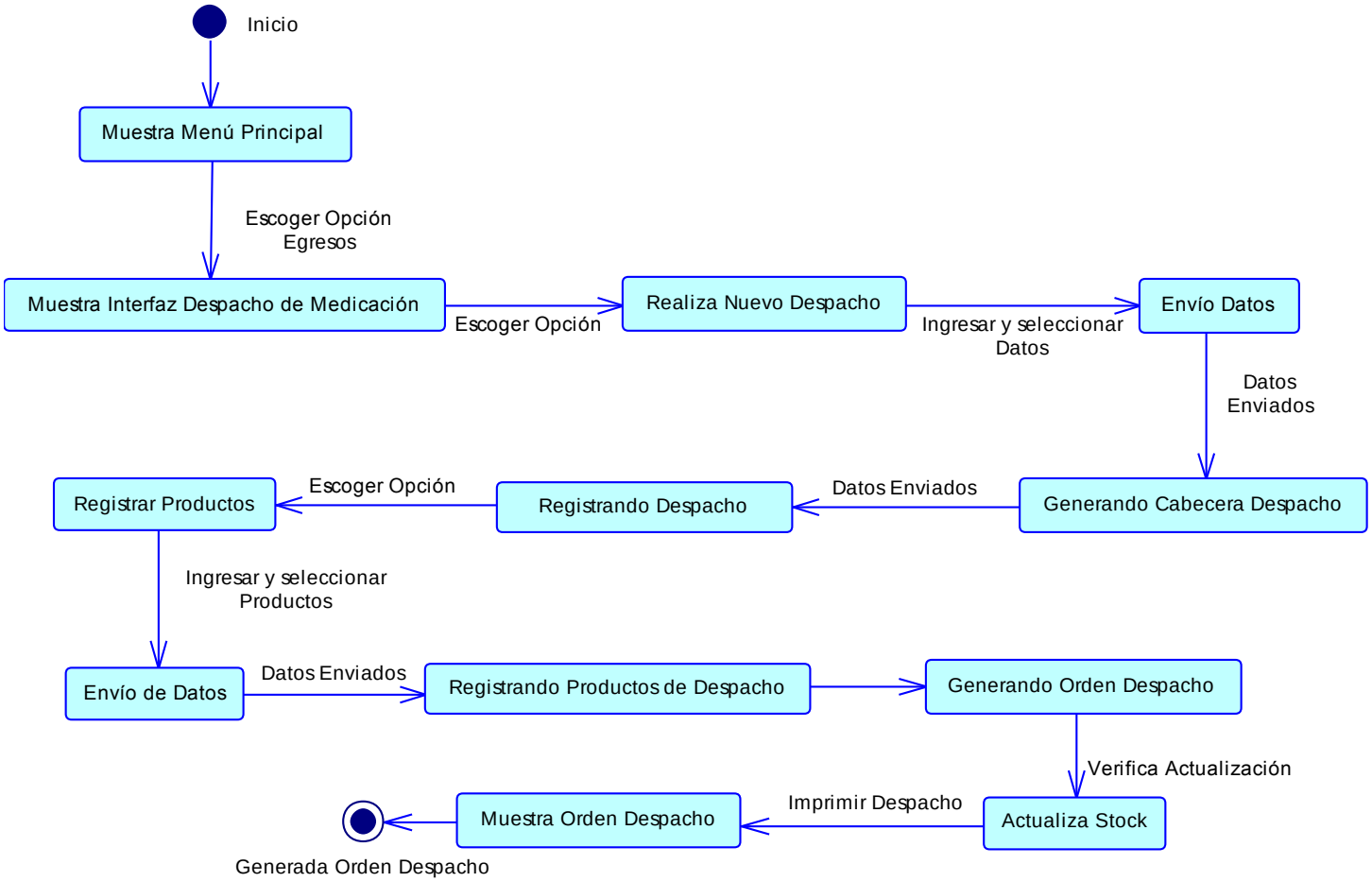


Figura. 3.3.1.35. Diagrama de Estado Realizar Despachos de Productos[62]

[62] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.36 Diagrama De Estado Modificar Despachos De Productos

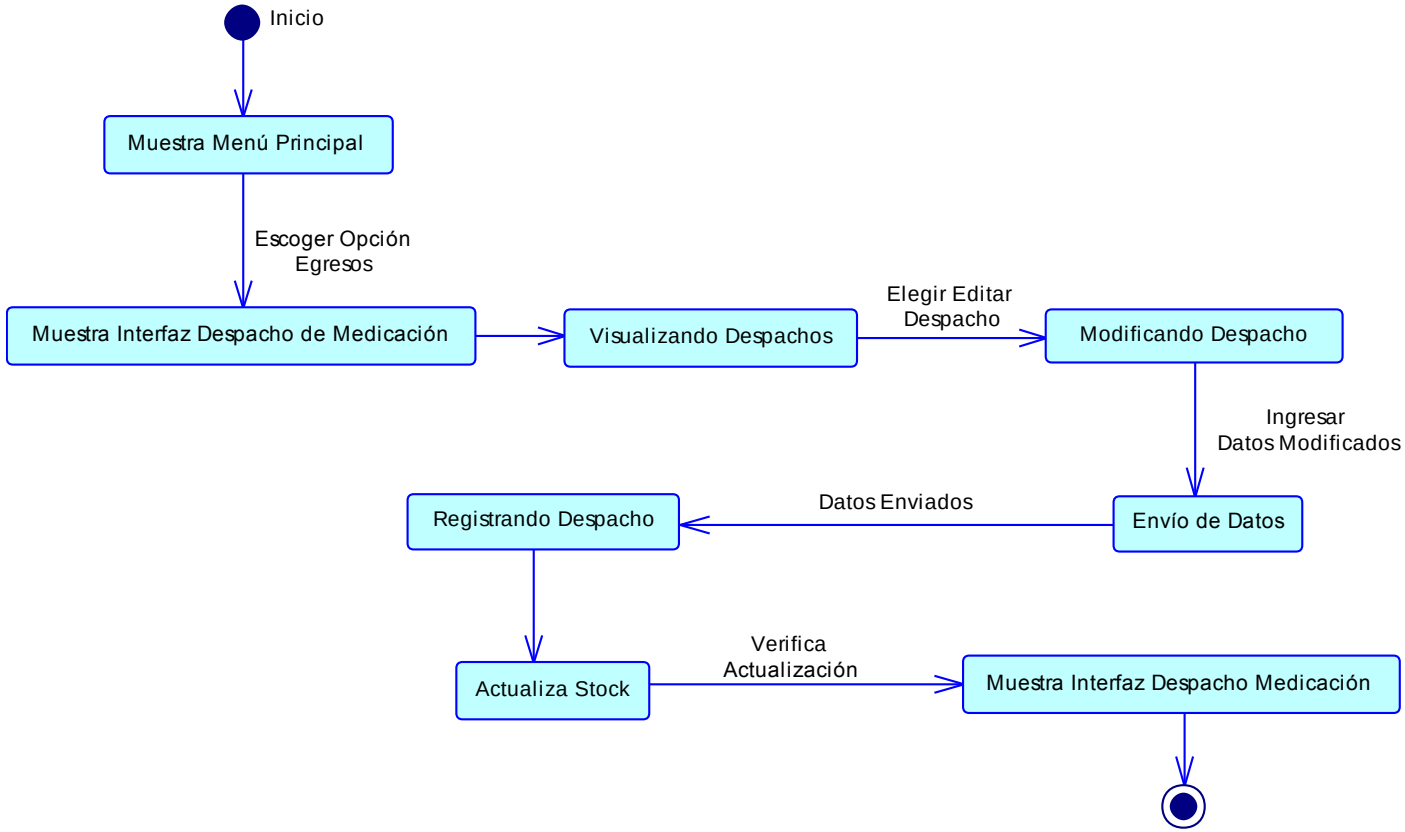


Figura. 3.3.1.36. Diagrama de Estado Modificar Despachos de Productos[63]

[63] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.3.1.37 Diagrama De Estado Eliminar Despachos De Productos

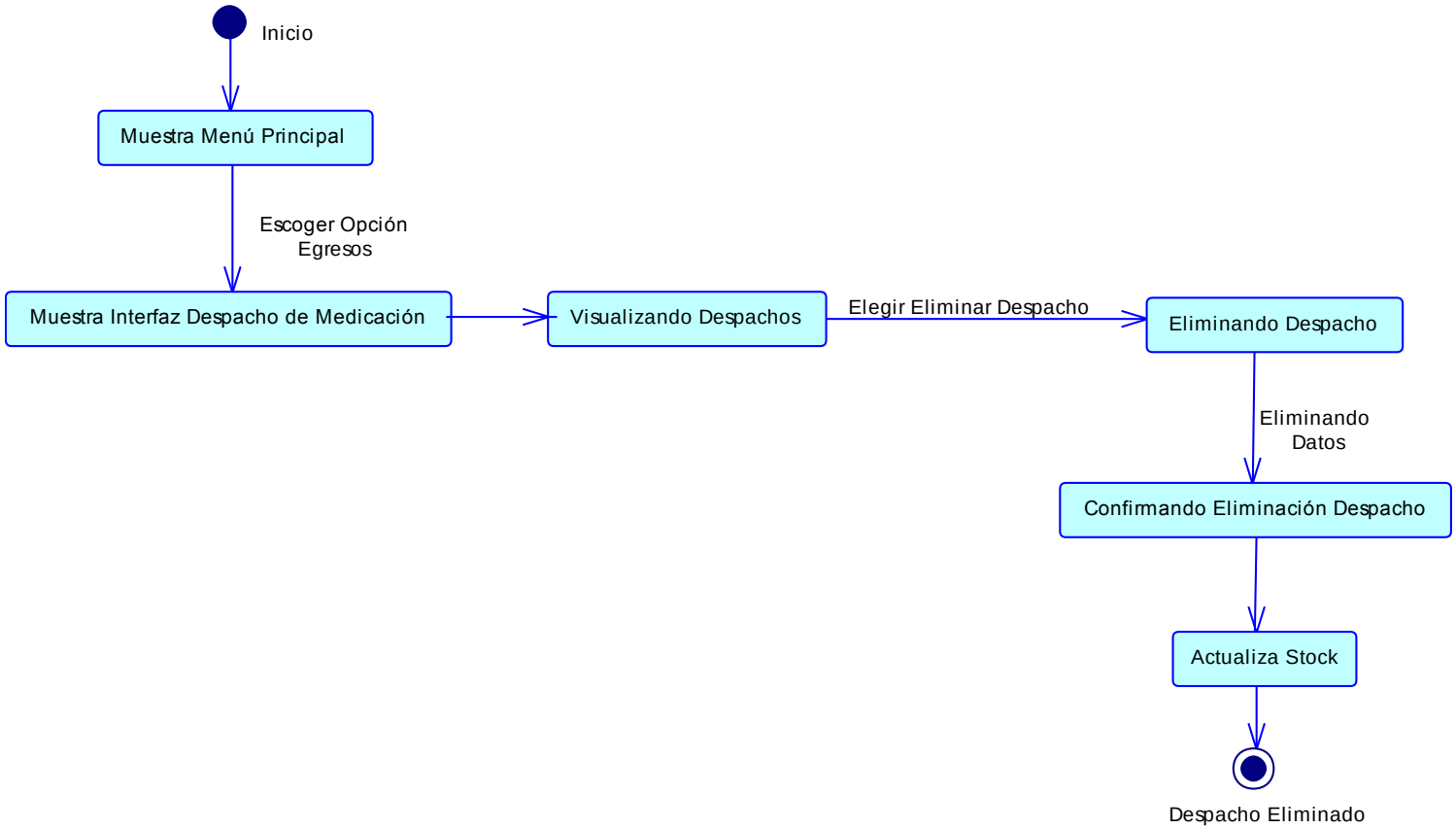


Figura. 3.3.1.37. Diagrama de Estado Eliminar Despachos de Productos[64]

[64] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4 DISEÑO DE LA BASE DE DATOS.

Una base de datos correctamente diseñada permite obtener acceso a información exacta y actualizada.

Un buen diseño de base de datos es aquel que:

- Divide la información en tablas basadas en temas para reducir los datos redundantes, los cuales malgastan el espacio y aumentan la probabilidad de que se produzcan errores e incoherencias.
- Proporciona a la base de datos creada la información necesaria para reunir la información de las tablas cuando así se precise.
- Ayuda a garantizar la exactitud e integridad de la información.
- Satisface las necesidades de procesamiento de los datos y de generación de informes.

Es por tal razón que el diseño de la base de datos del sistema SGMAS con el módulo de Gestión de Medicamentos se la realizó bajo un exhaustivo análisis de requerimientos y depuración minuciosa de la misma, logrando tener una base de datos óptima y eficiente para la realización exitosa del sistema.

3.4.1 MODELO LÓGICO DE LA BASE DE DATOS

El objetivo del diseño lógico es convertir los esquemas conceptuales locales en un esquema lógico global que se ajuste al modelo de SGBD[65] sobre el que se vaya a implementar el sistema. Mientras que el objetivo fundamental del diseño conceptual es la expresividad de los esquemas conceptuales locales, el objetivo del diseño lógico es obtener una representación que use, del modo más eficiente posible,

[65] *SGBD Sistemas de Gestión de Bases de Datos*

los recursos que el modelo de SGBD posee para estructurar los datos y para modelar las restricciones.

La metodología que se va a seguir para el diseño lógico en el modelo relacional consta de dos fases, cada una de ellas compuesta por varios pasos que se detallan a continuación.

En la primera fase, se construyen los esquemas lógicos locales para cada vista de usuario y se validan. En esta fase se refinan los esquemas conceptuales creados durante el diseño conceptual, eliminando las estructuras de datos que no se pueden implementar de manera directa sobre el modelo que soporta el SGBD, el modelo relacional. Una vez hecho esto, se obtiene un primer esquema lógico que se valida mediante la normalización y frente a las transacciones que el sistema debe llevar a cabo. El esquema lógico ya validado se puede utilizar como base para el desarrollo de prototipos. Una vez finalizada esta fase, se dispone de un esquema lógico para cada vista de usuario que es correcto, comprensible y sin ambigüedad.

Para el módulo de Gestión de Medicamentos este proceso se llevó a cabo en 3 meses dando una depuración óptima de la Base de Datos de la aplicación pudiendo identificar muy bien las tablas con las que se va a trabajar.

En este paso, se eliminan relaciones de muchos a muchos, las relaciones entre tres o más entidades, las relaciones recursivas, las relaciones redundantes, las relaciones con atributos, los atributos multievaluados, sustituyendo cada uno de ellos por una nueva entidad (débil) y una relación binaria de uno a muchos con la entidad original.

En el caso de las tablas de tb_producto y tb_lote se nota que la mayoría de las relaciones de las demás tablas sobre caen en las nombradas tablas, mientras que en las tablas que se relacionan con la

de tb_cantidad_lote se rompe una relación de varios a varios para tenerla como única tabla que relaciona productos con lote.

Una vez finalizado este paso, es más correcto referirse a los esquemas conceptuales locales refinados como esquemas lógicos locales, ya que se adaptan al modelo de base de datos que soporta el SGBD escogido.

En este paso, se obtiene un conjunto de relaciones (tablas) para cada uno de los esquemas lógicos locales en donde se representen las entidades y relaciones entre entidades. Cada relación de la base de datos tendrá un nombre, y el nombre de sus atributos aparecerá, a continuación, entre paréntesis. El atributo o atributos que forman la clave primaria se subrayan. Las claves ajenas, mecanismo que se utiliza para representar las relaciones entre entidades en el modelo relacional, se especifican aparte indicando la relación (tabla) a la que hacen referencia.

La normalización se utiliza para mejorar el esquema lógico, de modo que satisfaga ciertas restricciones que eviten la duplicidad de datos, garantice que el esquema resultante se encuentra más próximo al modelo del proyecto, que es consistente y que tiene la mínima redundancia y la máxima estabilidad.

Uno de los conceptos básicos del modelo relacional es que los atributos se agrupan en relaciones (tablas) porque están relacionados a nivel lógico.

El objetivo ahora es conseguir una base de datos normalizada por las siguientes razones:

- Un esquema normalizado organiza los datos de acuerdo a sus dependencias funcionales, es decir, de acuerdo a sus relaciones lógicas.

- La normalización obliga a entender completamente cada uno de los atributos que se han de representar en la base de datos.

- Un esquema normalizado es robusto y carece de redundancias, por lo que está libre de ciertas anomalías que éstas pueden provocar cuando se actualiza la base de datos.

- La normalización produce bases de datos con esquemas flexibles que pueden extenderse con facilidad.

Todas las restricciones de integridad establecidas en este paso se deben reflejar en el diccionario de datos para que puedan ser tenidas en cuenta durante la fase del diseño físico.

Un diagrama de flujo de datos muestra cómo se mueven los datos en la Institución y las áreas en donde se almacenan. Si se han utilizado diagramas de flujo de datos para modelar las especificaciones de requisitos de usuario, se pueden utilizar para comprobar la consistencia y completitud del esquema lógico desarrollado.

Los esquemas lógicos locales obtenidos hasta este momento se integrarán en un solo esquema lógico global en la siguiente fase para modelar los datos de todo la base de datos del módulo de gestión de medicamentos de nuestra aplicación.

SGMAS – Gestión de Medicamentos es un sistema grande, con varias vistas de usuario así como entidades y relaciones, no es relativamente sencillo comparar los esquemas locales, mezclarlos y resolver cualquier tipo de diferencia que pueda existir. Pero se debe seguir un proceso más sistemático para llevar a cabo este paso con éxito con esto podemos conseguir que nuestra base de datos quede totalmente depurada y su trabajo sea más funcional y eficaz:

- Revisar los nombres de las entidades y sus claves primarias.
- Revisar los nombres de las relaciones.
- Mezclar las entidades de las vistas locales.
- Incluir (sin mezclar) las entidades que pertenecen a una sola vista de usuario.
- Mezclar las relaciones de las vistas locales.
- Incluir (sin mezclar) las relaciones que pertenecen a una sola vista de usuario.
- Comprobar que no se ha omitido ninguna entidad ni relación.
- Comprobar las claves ajenas.
- Comprobar las restricciones de integridad.
- Dibujar el esquema lógico global.
- Actualizar la documentación.

3.4.1.1 Validar el esquema lógico global.

Este proceso de validación se realiza, de nuevo, mediante la normalización y mediante la prueba frente a las transacciones de los usuarios. Pero ahora sólo hay que normalizar las relaciones que hayan cambiado al mezclar los esquemas lógicos locales y sólo hay que probar las transacciones que requieran acceso a áreas que hayan sufrido algún cambio.

3.4.1.2 Estudiar el crecimiento futuro.

En este paso, se trata de comprobar que el esquema obtenido puede acomodar los futuros cambios en los requisitos con un impacto mínimo. Si el esquema lógico se puede extender fácilmente, cualquiera de los cambios previstos se podrá incorporar al mismo con un efecto mínimo sobre los usuarios existentes.

3.4.1.3 Revisar el esquema lógico global con los usuarios.

Una vez más, se debe revisar con los usuarios el esquema global y la documentación obtenida para asegurarse de que son una fiel representación de la Institución.

3.4.2 MODELO FÍSICO DE LA BASE DE DATOS

Uno de los objetivos principales del diseño físico es almacenar los datos de modo eficiente. Para medir la eficiencia hay varios factores que se deben tener en cuenta:

- Productividad de transacciones. Es el número de transacciones que se quiere procesar en un intervalo de tiempo.

- Tiempo de respuesta. Es el tiempo que tarda en ejecutarse una transacción. Desde el punto de vista del usuario, este tiempo debería ser el mínimo posible.

- Espacio en disco. Es la cantidad de espacio en disco que hace falta para los ficheros de la base de datos. Normalmente, el diseñador querrá minimizar este espacio.

Para mejorar las prestaciones, el diseñador del esquema físico debe saber cómo interactúan los dispositivos involucrados y cómo esto afecta a las prestaciones:

- Memoria principal. Los accesos a memoria principal son mucho más rápidos que los accesos a memoria secundaria, es aconsejable tener al menos un 5% de la memoria disponible, pero no más de un 10%. Si no hay bastante memoria

disponible para todos los procesos, el sistema operativo debe transferir páginas a disco para liberar memoria (paging). Cuando estas páginas se vuelven a necesitar, hay que volver a traerlas desde el disco. A veces, es necesario llevar procesos enteros a disco (swapping) para liberar memoria. El hacer estas transferencias con demasiada frecuencia empeora las prestaciones.

- CPU. La CPU controla los recursos del sistema y ejecuta los procesos de usuario. El principal objetivo con este dispositivo es lograr que no haya bloqueos de procesos para conseguirla. Si el sistema operativo, o los procesos de los usuarios, hacen muchas demandas de CPU, ésta se convierte en un cuello de botella.

- Entrada/salida a disco. Los discos tienen una velocidad de entrada/salida. Cuando se requieren datos a una velocidad mayor que ésta, el disco se convierte en un cuello de botella. Dependiendo de cómo se organicen los datos en el disco, se conseguirá reducir la probabilidad de empeorar las prestaciones.

- Red. La red se convierte en un cuello de botella cuando tiene mucho tráfico y cuando hay muchas colisiones.

Cada uno de estos recursos afecta a los demás, de modo que una mejora en alguno de ellos puede provocar mejoras en otros.

Para realizar un buen diseño físico es necesario conocer las consultas y las transacciones que se van a ejecutar sobre la base de datos. Esto incluye tanto información cualitativa, como cuantitativa. Para cada transacción, hay que especificar:

- La frecuencia con que se va a ejecutar.

- Las relaciones y los atributos a los que accede la transacción, y el tipo de acceso: consulta, inserción, modificación o eliminación. Los atributos que se modifican no son buenos candidatos para construir estructuras de acceso.

- Los atributos que se utilizan en los predicados del WHERE de las sentencias SQL. Estos atributos pueden ser candidatos para construir estructuras de acceso dependiendo del tipo de predicado que se utilice.

- Si es una consulta, los atributos involucrados en el join de dos o más relaciones. Estos atributos pueden ser candidatos para construir estructuras de acceso.

- Las restricciones temporales impuestas sobre la transacción. Los atributos utilizados en los predicados de la transacción pueden ser candidatos para construir estructuras de acceso.

En ocasiones puede ser conveniente relajar las reglas de normalización introduciendo redundancias de forma controlada, con objeto de mejorar las prestaciones del sistema. Por regla general, la desnormalización de una relación puede ser una opción viable cuando las prestaciones que se obtienen no son las deseadas y la relación se actualiza con poca frecuencia, pero se consulta muy a menudo. Las redundancias que se pueden incluir al desnormalizar son de varios tipos: se pueden introducir datos derivados (calculados a partir de otros datos), se pueden duplicar atributos o se pueden hacer joins de relaciones.

No se pueden establecer una serie de reglas que determinen cuándo desnormalizar relaciones, pero hay algunas situaciones muy comunes en donde puede considerarse esta posibilidad:

Combinar relaciones de uno a uno. Cuando hay relaciones (tablas) involucradas en relaciones de uno a uno, se accede a ellas de manera conjunta con frecuencia y casi no se les accede separadamente, se pueden combinar en una sola relación (tabla).

- Duplicar atributos no clave en relaciones de uno a muchos para reducir los joins. Para evitar operaciones de join, se pueden incluir atributos de la relación (tabla) padre en la relación (tabla) hijo de las relaciones de uno a muchos.

- Duplicar claves ajenas en relaciones de uno a muchos para reducir los joins. Para evitar operaciones de join, se pueden incluir claves ajenas de una relación (tabla) en otra relación (tabla) con la que se relaciona (habrá que tener en cuenta ciertas restricciones).

- Duplicar atributos en relaciones de muchos a muchos para reducir los joins.
- Durante el diseño lógico se eliminan las relaciones de muchos a muchos introduciendo dos relaciones de uno a muchos. Esto hace que aparezca una nueva relación (tabla) intermedia, de modo que si se quiere obtener la información de la relación de muchos a muchos, se tiene que realizar el join de tres relaciones (tablas). Para evitar algunos de estos joins se pueden incluir algunos de los atributos de las relaciones (tablas) originales en la relación (tabla) intermedia.

- Introducir grupos repetitivos. Los grupos repetitivos se eliminan en el primer paso de la normalización para conseguir la

primera forma normal. Estos grupos se eliminan introduciendo una nueva relación (tabla), generando una relación de uno a muchos. A veces, puede ser conveniente reintroducir los grupos repetitivos para mejorar las prestaciones.

- Todas las redundancias que se introduzcan en este paso se deben documentar y razonar. El esquema lógico se debe actualizar para reflejar los cambios introducidos.

En caso de que se tenga que adquirir nuevo equipamiento informático, el diseñador debe estimar el espacio necesario en disco para la base de datos. Esta estimación depende del SGBD que se vaya a utilizar y del hardware. En general, se debe estimar el número de tuplas de cada relación y su tamaño. También se debe estimar el factor de crecimiento de cada relación.

3.4.3 DICCIONARIO DE DATOS.

El diccionario de datos contiene las características lógicas de los datos que se van a utilizar en el Módulo de Gestión de Medicamentos del Sistema de Gestión Médico para Áreas de Salud – SGMAS, de los datos que se desarrollan durante el análisis de flujo de datos y los requerimientos del sistema.

Por lo general, el diccionario de datos está integrado en el sistema de información el cual detallamos el contenido cada tabla con sus respectivos campos, tipos, relaciones de la base de datos utilizada para el desarrollo del módulo.

INTERPRETACIÓN DE TABLAS DEL DICCIONARIO DE DATOS

No.	TABLA	INTERPRETACIÓN
3.4.3.1	TABLA BAJAS	Tabla en la que se guardan medicamentos que por algún tipo de siniestro (inundación, terremoto, etc.) sufrieron daños irreparables y que es necesario darlos de baja.
3.4.3.2	TABLA BODEGA	Tabla en la que se guardan los datos generales de la bodega en la que se van almacenar los medicamentos.
3.4.3.3	TABLA CANTIDAD LOTE	Tabla en la que se guardan la cantidad de medicamentos (stock) que llegan a bodega.
3.4.3.4	TABLA DOCUMENTO	Tabla en la que se guardan los datos detallados del documento contable que ingresa a bodega (nota de venta, factura, memorando, etc.).
3.4.3.5	TABLA FAMILIA	Tabla en la que se guardan los datos de los productos ingresan a bodega dependiendo de su característica (insumos médicos o medicamentos).
3.4.3.6	TABLA MOVIMIENTO	Tabla en la que se guardan los datos de la cabecera para la realización de los despachos de medicamentos.
3.4.3.7	TABLA DETALLE MOVIMIENTO	Tabla en la que se guardan cada uno de los medicamentos de los a ser despachados en detalle.
3.4.3.8	TABLA LOTE	Tabla en la que se guardan todos los medicamentos que ingresan a bodega para posteriormente despacharlos.
3.4.3.9	TABLA PRESENTACIÓN	Tabla en la que se guardan la presentación que tiene cada medicamento a ser almacenado (envases, tabletas, spray, etc.).

3.4.3.10	TABLA PRODUCTO	Tabla en la que se guardan los datos generales de los medicamentos a ser ingresados en la tabla de lotes de medicación.
3.4.3.11	TABLA PROVEEDORES	Tabla en la que se guardan datos generales de los proveedores de la medicación que ingresa a bodega para su almacenamiento.
3.4.3.12	TABLA RESPONSABLE	Tabla en la que se guardan los datos de del responsable de la bodega en el cual se va almacenar los medicamentos.
3.4.3.13	TABLA TIPO DOCUMENTO	Tabla en la que se guardan los datos del tipo de documento contable que utilicen en el proceso de despacho de medicamentos (nota de venta, factura, memorando, etc.).
3.4.3.14	TABLA UNIDADES	Tabla en la se guardan los datos de la unidad de medida del medicamento ingresado (10 ml, 100 gr, etc.).

Tabla 3.4.3 Interpretación de tablas del Diccionario de Datos. [66]

[66] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.1 Tabla Bajas

Tabla: tb_bajas (Permite ingresar los productos y/o medicamentos que son dados de baja)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_lote	Código de lote	INT8		S	PK, FK	NOT NULL
id_producto	Código producto	INT8		S	PK, FK	NOT NULL
cantidad_baja	Cantidad de medicamentos dados de baja	INT8		S		NOT NULL
fecha_baja	Fecha de ingreso de la baja	DATE		S		NOT NULL
razon_baja	Motivo de la baja	VARCHAR	100	S		NOT NULL
Clave Primaria	PK_LOTE(id_lote)					
Clave Foraneas	tb_lote - tb_bajas tb_producto - tb_bajas					
Claves Únicas						

Tabla 3.4.3.1: Diccionario de Datos: Tabla Bajas.[67]

[67] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.2 Tabla Bodega

Tabla: tb_bodega (Permite ingresar datos de la bodega)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_bodega	Código de bodega *	INT8		S	PK	NOT NULL
nombre_bodega	Nombre de la bodega donde se almacena el medicamento	VARCHAR	50	S		NOT NULL
direccion_bodega	Dirección de la bodega donde se almacena el medicamento	VARCHAR	100	S		NOT NULL
telefono_bodega	Teléfono de la bodega donde se almacena el medicamento	VARCHAR	20	S		NOT NULL
Clave Primaria	PK_BODEGA(id_bodega)					
Clave Foraneas						
Claves Únicas						

Tabla 3.4.3.2: Diccionario de Datos: Tabla Bodega.[68]

[68] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.3 Tabla Cantidad de Lote

Tabla: tb_cantidad_lote (Permite ingresar cantidad de medicamento por Lote)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_lote	Código lote	VARCHAR	20	S	PK, FK	NOT NULL
id_producto	Código producto	INT8		S	PK, FK	NOT NULL
cantidad_lote	Cantidad de medicamento existente en bodega	INT8		S		NOT NULL
Clave Primaria	PK_LOTE(CODIGO_TUR)					
Clave Foraneas	tb_lote - tb_cantidad_lote tb_producto - tb_cantidad_lote					
Claves Únicas						

Tabla 3.4.3.3: Diccionario de Datos: Tabla Cantidad de Lote.[69]

[69] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.4 Tabla Documento

Tabla: tb_documento (Permite ingresar detalle del Documento Contable)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_doc	Código del documento contable*	INT8		S	PK	NOT NULL
ruc_prov	Código del proveedor	INT8		S	FK	NOT NULL
id_tipo_doc	Código del tipo de documento contable (factura, nota de venta, etc.)	INT8		S	FK	NOT NULL
fecha_doc	Fecha de emisión del documento contable	DATE		S		NOT NULL
valor_doc	Monto total del documento contable	DECIMAL	(8,0)	S		NOT NULL
num_ref_doc	Número de referencia del documento contable	VARCHAR	20	S		NOT NULL
Clave Primaria	PK_DOCUMENTO(id_doc)					
Clave Foraneas	tb_proveedores - tb_documento tb_tipo_doc - tb_documento					
Claves Únicas						

Tabla 3.4.3.4: Diccionario de Datos: Tabla Detalle Documento Contable.[70]

[70] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.5 Tabla Familia

Tabla: tb_familia (Permite ingresar datos de Tipo de Familia)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Resticción Validación**
id_familia	Código de la familia*	INT8		S	PK	NOT NULL
nombre_familia	Nombre de la familia dependiendo del almacenamiento en bodega (medicamentos, insumos médicos, etc.)	VARCHAR	50	S		NOT NULL
observaciones_familia	Observaciones referentes al caso	VARCHAR	100	S		NOT NULL
Clave Primaria	PK_FAMILIA(id_familia)					
Clave Foraneas						
Claves Únicas						

Tabla 3.4.3.5: Diccionario de Datos: Tabla Familia.[71]

[71] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.6 Tabla Movimiento

Tabla: tb_movimiento (Permite ingresar datos del Movimiento de los productos y/o medicamentos)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_movimiento	Código del movimiento*	INT8		S	PK	NOT NULL
id_bodega	Código de la bodega donde se realiza el movimiento	INT8		S	FK	NOT NULL
id_transaccion	Código de transacción	INT8		S	FK	NOT NULL
responsable_movimiento	Responsable de la realización del movimiento	VARCHAR	50	S		NOT NULL
fecha_movimiento	Fecha de realización del movimiento	DATE		S		NOT NULL
estado_movimiento	Se almacena el estado del movimiento dependiendo del pedido (despachado, pendiente, negado)	VARCHAR	20	S		NOT NULL
tipo_trans_movimiento	Tipo de transacción dependiendo del código de transacción al que pertenece (egreso de bodega, pedido, etc.)	VARCHAR	50	S		NOT NULL
tipo_pedido		VARCHAR	50	S		NOT NULL
Clave Primaria	PK_MOVIMIENTO(id_movimiento)					
Clave Foraneas	tb_bodega - tb_movimiento tb_transaccion - tb_movimiento					
Claves Únicas						

Tabla 3.4.3.6: Diccionario de Datos: Tabla Movimiento.[72]

[72] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.7 Tabla Detalle de Movimiento

Tabla: tb_mov_detalle (Permite ingresar el detalle del Movimiento)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Resticción Validación**
id_movimiento	Código del movimiento	INT8		S	PK,FK	NOT NULL
id_lote	Código del lote	VARCHAR	20	S	PK,FK	NOT NULL
id_producto	Código del producto	INT8		S	PK,FK	NOT NULL
cantidad_mov_prod	Cantidad de medicación solicitada en el movimiento	INT8		S		NOT NULL
Clave Primaria	PK_MOVIMIENTO(id_movimiento)					
Clave Foraneas	tb_movimiento - tb_mov_detalle tb_producto - tb_mov_detalle tb_lote - tb_mov_detalle					
Claves Únicas						

Tabla 3.4.3.7: Diccionario de Datos: Tabla Detalle de Movimiento[73]

[73] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.8 Tabla Lote

Tabla: tb_lote (Permite ingresar datos del Lote de los productos y/o medicamentos)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_lote	Código del lote de medicación*	VARCHAR	20	S	PK	NOT NULL
id_producto	Código producto	INT8		S	PK,FK	NOT NULL
id_doc	Código de documento contable	INT8		S	PK,FK	NOT NULL
marca_lote	Marca o empresa encargada de su fabricación	VARCHAR	20	S		NOT NULL
descripcion_lote	Breve descripción del lote	VARCHAR	50	S		NOT NULL
recepción_lote	Fecha de recepción del lote de medicación	DATE		S		NOT NULL
elaboración_lote	Fecha de elaboración del lote de medicación	DATE		S		NOT NULL
caducidad_lote	Fecha de caducidad del lote de medicación	DATE		S		NOT NULL
costo_lote	Costo por lote de medicación	DECIMAL	(8,0)	S		NOT NULL
cantidad_lote	Cantidad recibida del medicamento	INT8		S		NOT NULL
Clave Primaria	PK_LOTE(id_lote)					
Clave Foraneas	tb_producto - tb_lote tb_doc - tb_lote					
Claves Únicas						

Tabla 3.4.3.8: Diccionario de Datos: Tabla Lote de Medicamentos.[74]

[74] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.9 Tabla Presentación

Tabla: tb_presentación (Permite ingresar datos de Presentación del producto y/o medicamento)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_presentacion	Código de la presentación del medicamento*	INT8		S	PK	NOT NULL
nombre_presentacion	Nombre de la presentación del medicamento a almacenar (envases, tabletas, frasco, spray, etc.)	VARCHAR	50	S		NOT NULL
observaciones_presentacion	Observaciones referentes al caso	VARCHAR	100	S		NOT NULL
Clave Primaria	PK_PRESENTACION(id_presentacion)					
Clave Foraneas						
Claves Únicas						

Tabla 3.4.3.9: Diccionario de Datos: Tabla Presentación.[75]

[75] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.10 Tabla Producto

Tabla: tb_producto (Permite ingresar datos de Producto)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_producto	Código producto*	INT8		S	PK	NOT NULL
id_bodega	Código de la bodega donde se encuentra almacenado el medicamento	INT8		S	FK	NOT NULL
id_familia	Código de la familia de la que pertenece le producto	INT8		S	FK	NOT NULL
id_presentacion	Código de la presentación que posee la medicación	INT8		S	FK	NOT NULL
id_unidad	Código de la unidad de medida del producto	INT8		S	FK	NOT NULL
nombre_producto	Nombre del medicamento	VARCHAR	100	S		NOT NULL
observación_producto	Se colocara alguna observación a la medicación ingresada	VARCHAR	100	S		NOT NULL
stock_max	Campo de control de exceso de stock	INT8		S		NOT NULL
stock_min	Campo de control de déficit de stock	INT8		S		NOT NULL
Clave Primaria	PK_PRODUCTO(id_producto)					
Clave Foraneas	tb_bodega- tb_producto tb_familia - tb_producto tb_presentacion - tb_producto tb_unidad - tb_producto					
Claves Únicas						

Tabla 3.4.3.10: Diccionario de Datos: Tabla Producto.[76]

[76] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.11 Tabla Proveedores

Tabla: tb_proveedores (Permite ingresar datos de Proveedores)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
ruc_prov	RUC o número de cédula del proveedor	INT8		S	PK	NOT NULL
nombre_prov	Nombre del proveedor que distribuye o entrega el medicamento	VARCHAR	50	S		NOT NULL
direccion_prov	Dirección del proveedor que distribuye o entrega el medicamento	VARCHAR	100	S		NOT NULL
telefono_prov	Teléfono del proveedor que distribuye o entrega el medicamento	VARCHAR	20	S		NOT NULL
responsable_prov	Campo para identificar el responsable de la entrega de la medicación	VARCHAR	50	S		NOT NULL
Clave Primaria	PK_PROV(ruc_prov)					
Clave Foraneas						
Claves Únicas						

Tabla 3.4.3.11: Diccionario de Datos: Tabla Proveedores.[77]

[77] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.12 Tabla Responsable

Tabla: tb_responsable (Permite ingresar datos de Responsable)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_responsable	Código responsable*	INT8		S	PK	NOT NULL
id_bodega	Código de la bodega que identifica el responsable de la misma	INT8		S	FK	NOT NULL
nombre_responsable	Nombre del responsable o encargado de la bodega	VARCHAR	50	S		NOT NULL
direccion_responsable	Dirección del responsable o encargado de la bodega	VARCHAR	100	S		NOT NULL
telefono_responsable	Teléfono del responsable o encargado de la bodega	VARCHAR	20	S		NOT NULL
mail_responsable	Mail del responsable o encargado de la bodega	VARCHAR	20			
Clave Primaria	PK_RESPONSABLE(id_responsable)					
Clave Foraneas	tb_bodega- tb_responsable					
Claves Únicas						

Tabla 3.4.3.12: Diccionario de Datos: Tabla Responsable.[78]

[78] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.13 Tabla Tipo Documento

Tabla: tb_tipo_doc (Permite ingresar Tipo de Documento)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_tipo_doc	Código tipo de documento contable*	INT8		S	PK	NOT NULL
tipo_doc	Tipo de documento contable (nota de venta, factura, recibo, etc.)	VARCHAR	50	S		NOT NULL
Clave Primaria	PK_TIPO_DOC(id_tipo_doc)					
Clave Foraneas						
Claves Únicas						

Tabla 3.4.3.13: Diccionario de Datos: Tabla Tipo de Documento Contable.[79]

[79] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

3.4.3.14 Tabla Unidades

Tabla: tb_unidades (Permite ingresar documento)						
Campo	**Descripción**	**Tipo-Dato**	**Extensión**	**Obligatorio**	**Tipo restricción**	**Restricción Validación**
id_unidad	Código de la unidad de medida*	INT8		S	PK	NOT NULL
unidad_medida	Unidad de medida dependiendo al medicamento	VARCHAR	50	S		NOT NULL
observaciones_unidad	Observaciones referentes al caso	VARCHAR	50	S		NOT NULL
Clave Primaria	PK_UNIDAD(id_unidad)					
Clave Foraneas						
Claves Únicas						

Tabla 3.4.3.14: Diccionario de Datos: Tabla Unidad de Medida de Medicamento.[80]

[80] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

4 CAPÍTULO IV DESARROLLO

4.1 CONSTRUCCIÓN DE FORMULARIOS

La construcción de los formularios en zend framework del sistema se los realiza en script.phtml con esto podemos tener el control de toda la maquetación del formulario así como de las validaciones de todos los campos que se coloquen en el formulario dependiendo de las necesidades del sistema siendo estas más sencillas y que requieren de un menor grado de seguridad, al ser algoritmos conocidos por todo el mundo, por ejemplo, validación de campo vacío, validación de e-mail, validación de que todos los caracteres del campo son numéricos, etc.

Se crea la clase Form donde seguiremos construyendo el formulario y añadiendo las validaciones necesarias que deban ser realizadas a nivel de servidor y así tendremos una vista donde pondremos el resultado final del proceso realizado.

Se utilizó javascripts para las correspondientes validaciones de todos los campos de los formularios del sistema de esta forma se crea un archivo con extensión .js en el directorio: /public/js/ de nuestro proyecto.

4.2 CONSTRUCCIÓN DE INTERFACES

Para la construcción de cada una de las interfaces del sistema se ha creado cada formulario en lenguaje PHP con su respectiva extensión que se guardan en la carpeta correspondiente a los forms en el proyecto de zend framework con el nombre de cada interfaz que compone el sistema debidamente identificados.

Uno de los métodos utilizados es el Zend_Form que es el componente de Zend Framework encargado de crear y manejar los formularios que usemos en nuestras aplicaciones para la construcción del sistema, por lo

que es un componente de uso casi obligado en cualquier desarrollo web y uno de los que más llaman la atención al empezar a trabajar con este framework.

Entre sus principales responsabilidades se encuentran:

- Validar los formularios y mostrar los errores
- Filtrado (Filter) de datos (escape / normalización de los datos recibidos)
- Generar el HTML Markup del form y sus elementos

Todo esto convierte a Zend_Form en una herramienta realmente simple y potente que nos permite ahorrar tiempo al trabajar con formularios y generar un código mucho más ordenado y simplificado.

Zend trae por default los siguientes tipos de elementos:

Button, checkbox / multiCheckbox, hidden, image, password, radio, reset, select / multi-select, submit, text, textarea.

Pero a su vez las validaciones que proporciona esta herramienta se los hace a nivel del servidor por lo que resulta una carga al mismo lo que nos lleva a la solución de aplicar como se dijo anteriormente la utilización de javascripts para realizar esa función pues a diferencia de lo antes expuesto estas validaciones se las hace a nivel del cliente.

El siguiente código es la construcción de la interfaz para el manejo de los medicamentos con las entradas de datos correspondientes a la información general cada uno de los productos ingresados al sistema.

Para el ejemplo se utiliza el método Zend_Form con sus correspondientes validaciones.

```
<?php
class Application_Form_Producto extends Zend_Form
{
public function init()
   {
$this->setName('producto');

// Campo hidden para guardar id del producto
        $id_producto = new Zend_Form_Element_Hidden('id_producto');
        $id_producto->addFilter('Int');
        $id_bodega = new Zend_Form_Element_Select('id_bodega');
        $id_bodega->setLabel('')->setRequired(true);

// Carga en un select los tipos de bodega existentes en Base de Datos
$table = new Application_Model_DbTable_Bodega();

//obtengo listado de todas las bodegas y los recorro en un
  //arreglo para agregarlos a la lista
    foreach ($table->listar() as $c)
            {
               $id_bodega->addMultiOption($c->id_bodega, $c-
    >nombre_bodega);
            }
            $id_familia = new Zend_Form_Element_Select('id_familia');
            $id_familia->setLabel('')->setRequired(true);
            $table2 = new Application_Model_DbTable_Familia();
    foreach ($table2->listar() as $c1)
    {
               $id_familia->addMultiOption($c1->id_familia, $c1-
    >nombre_familia);
    }
```

```
        $id_presentacion = new
Zend_Form_Element_Select('id_presentacion');
        $id_presentacion->setLabel('')->setRequired(true);
        $table3 = new Application_Model_DbTable_Presentacion();
foreach ($table3->listar() as $c2)
{
            $id_presentacion->addMultiOption($c2->id_presentacion, $c2-
>nombre_presentacion);
}
        $id_unidad = new Zend_Form_Element_Select('id_unidad');
        $id_unidad->setLabel('')->setRequired(true);
        $table4 = new Application_Model_DbTable_Unidades();
foreach ($table4->listar() as $c3)
{
            $id_unidad->addMultiOption($c3->id_unidad, $c3-
>unidad_medida);
}

        $id_comercial = new Zend_Form_Element_Select('id_comercial');
        $id_comercial->setLabel('')->setRequired(true);
       $table5 = new Application_Model_DbTable_Comercial();
foreach ($table5->listar() as $c4)
{
         $id_comercial->addMultiOption($c4->id_comercial,$c4-
>presentacion_comercial);
        }

        //creamos <input text> para escribir nombre album
        $nombre_producto = new
Zend_Form_Element_Text('nombre_producto');

$nombre_producto->setLabel('Nombre del producto:')

        //VALIDACIONES DE ZEND
```

```
->setRequired(true) // CAMPO OBLIGATORIO
            ->addFilter('StripTags')
->addFilter('StringTrim') // ELIMINACIÓN DE CAMPOS BLANCO
            ->addFilter('StringToUpper') // CONVERSION MAYÚSCULAS
            ->addValidator('NotEmpty'); // CAMPO NO DEBE ESTAR EN
                                        // BLANCO

        $estado_producto = new
Zend_Form_Element_Text('estado_producto');
        $estado_producto->setLabel('')
->setRequired(false)
            ->addFilter('StripTags')
            ->addFilter('StringToUpper')
            ->addFilter('StringTrim');

      $observacion_producto = new
        Zend_Form_Element_Text('observacion_producto');
      $observacion_producto->setLabel('')
            ->setRequired(false)
            ->addFilter('StripTags')
            ->addFilter('StringToUpper')
            ->addFilter('StringTrim');

      $stock_max = new Zend_Form_Element_Text('stock_max');
      $stock_max->setLabel('')
            ->setRequired(false)
            ->addFilter('StripTags')
            ->addValidator('Digits')
            ->addValidator('NotEmpty')
            ->addFilter('StringTrim');

      $stock_min = new Zend_Form_Element_Text('stock_min');
      $stock_min->setLabel('')
```

```
                ->setRequired(false)
                ->addFilter('StripTags')
                ->addValidator('Digits')
                ->addValidator('NotEmpty')
  ->addFilter('StringTrim');


        //botón para enviar formulario
          $submit = new Zend_Form_Element_Submit('submit');
          $submit->setAttrib('id', 'submitbutton');

        $this->addElements(array($id_producto, $id_bodega,
          $id_familia, $id_presentacion, $id_unidad,
          $id_comercial,$nombre_producto, $estado_producto,
  $observacion_producto, $stock_max, $stock_min, $submit));
  }
  }
```

Con este método se le carga al servidor todas las validaciones del formulario que se pueden hacer a nivel de cliente así como la imposibilidad de crear formularios cuyos campos no sean secuenciales.

De la misma forma no se puede personalizar los formularios con la utilización de "decorators" que nos permiten "customizar" los formularios y darle el estilo que queramos, en muchas ocasiones significa introducir código HTML en el código de servidor; es por tal motivo que nuestro objetivo siempre es separar claramente las capas de negocio y la de presentación, y aunque esta segunda tenga la clase "Form" en ella no es aconsejable introducir código que pertenezca a la vista. Por tal motivo realizamos cambios en la vista del proyecto e introducimos código HTML de manera que podamos modificar y manipular el estilo del formulario a la medida que los formularios sean de fácil manipulación para el usuario y para el eficiente trabajo del servidor brindándole más agilidad para la resolución de las actividades para lo que fue encomendado.

Para ello modificamos el archivo que zend crea simultáneamente en el directorio: /views/producto, con el mismo nombre del Form creado introduciendo lenguaje HTML y creando archivos javascripts para la validación de los campos del mismo formulario.

En el siguiente código vemos cómo podemos realizar formularios modificando la vista del directorio: /views/producto/add.phtml, así:

```
<h3><center><?php echo $this->escape($this->title);
?></center></h3><BR>

<center>
<div id="fondogrid">
<form action="" id="producto" onsubmit="return validatefields();"
method="post" name="producto">

<p style="color: red; font-family: bold, Arial; font-size: 16px">* Campos
Obligatorios</p>

<fieldset><legend>Datos del Medicamento</legend><br>

<table style="text-align: left; width: 765px; height: 80px; vertical-align:
middle;"
border="0">
<tbody>
<tr>
<td>
<label>CUM:</label><label style="color: red">*</label>
</td>
<td>
<input name="observacion_producto" id="observacion_producto"
class="mayusculas" size="20" type="text"/>
</td>
```

```
</tr>
<tr>
<td>
<label>NOMBRE DE LA BODEGA:</label><label style="color:
red">*</label>
</td>
<td>
<select name="id_bodega" id="id_bodega" class="listc">
<option value="0">[Seleccione opcion]</option>
<?php foreach ($this->bodega as $c) : ?>
<?php if ($c->id_bodega == $this->id_bodega) { ?>
<option value="<?php echo $this->escape($c->id_bodega); ?>"
selected><?php echo $this->escape($c->nombre_bodega); ?></option>
<?php }else { ?>
<option value="<?php echo $this->escape($c->id_bodega); ?>"><?php
echo $this->escape($c->nombre_bodega); ?></option>
<?php } ?>
<?php endforeach; ?>
</select>
</td>
<td>
<label>FAMILIA DEL PRODUCTO:</label><label style="color:
red">*</label>
</td>
<td>
<select name="id_familia" id="id_familia" class="listc">
<option value="0">[Seleccione opcion]</option>
<?php foreach ($this->familia as $c) : ?>
<?php if ($c->id_familia == $this->id_familia) { ?>
<option value="<?php echo $this->escape($c->id_familia); ?>"
selected><?php echo $this->escape($c->nombre_familia); ?></option>
<?php }else { ?>
<option value="<?php echo $this->escape($c->id_familia); ?>"><?php
echo $this->escape($c->nombre_familia); ?></option>
```

```
<?php } ?>
<?php endforeach; ?>
</select>
</td>
</tr>
<tr>
<td>
<label>FORMA FARMACEUTICA:</label><label style="color:
red">*</label>
</td>
<td>
<select name="id_presentacion" id="id_presentacion" class="listc">
<option value="0">[Seleccione opcion]</option>
<?php foreach ($this->presentacion as $c) : ?>
<?php if ($c->id_presentacion == $this->id_presentacion) { ?>
<option value="<?php echo $this->escape($c->id_presentacion); ?>"
selected><?php echo $this->escape($c->nombre_presentacion);
?></option>
<?php }else { ?>
<option value="<?php echo $this->escape($c->id_presentacion);
?>"><?php echo $this->escape($c->nombre_presentacion); ?></option>
<?php } ?>
<?php endforeach; ?>
</select
</td>
<td>
<label>CONCENTRACION:</label><label style="color: red">*</label>
</td>
<td>
<select name="id_unidad" id="id_unidad" class="listc">
<option value="0">[Seleccione opcion]</option>
<?php foreach ($this->unidad as $c) : ?>
<?php if ($c->id_unidad == $this->id_unidad) { ?>
```

```
<option value="<?php echo $this->escape($c->id_unidad); ?>"
selected><?php echo $this->escape($c->unidad_medida); ?></option>
<?php }else { ?>
<option value="<?php echo $this->escape($c->id_unidad); ?>"><?php
echo $this->escape($c->unidad_medida); ?></option>
<?php } ?>
<?php endforeach; ?>
</select
</td>

</tr>
<tr>
<td>
<label>NOMBRE DEL PRODUCTO:</label><label style="color:
red">*</label>
</td>
<td>
<input id="nombre_producto" class="mayusculas" maxlength="20"
size="20" tabindex="3" name="nombre_producto">
</tr>
<tr>
<td>
<label>ESTADO DEL PRODUCTO:</label><label style="color:
red">*</label>
</td>
<td>
<select name="estado_producto" id="estado_producto">
<option value="NINGUNO">[Seleccione Estado]</option>
<option value="RESAGADO">RESAGADO</option>
<option value="DESCONTINUADO">DESCONTINUADO</option>
<option value="DETERIORADO">DETERIORADO</option>
</select>
<!--<input id="estado_producto" maxlength="20" size="20" tabindex="3"
name="estado_producto">-->
```

```
</td>
</tr>
</tbody>
</table>
<table style="text-align: left; width: 765px; vertical-align: middle;"
border="0">
<tbody>
<tr>
<td>
<label>PRESENTACIÓN COMERCIAL:</label><label style="color:
red">*</label>
</td>
<td>
<select name="id_comercial" id="id_comercial" class="listc" style="text-
align: left;">
<option value="0">[Seleccione opcion]</option>
<?php foreach ($this->comercial as $c) : ?>
<?php if ($c->id_comercial == $this->id_comercial) { ?>
<option value="<?php echo $this->escape($c->id_comercial); ?>"
selected><?php echo $this->escape($c->presentacion_comercial);
?></option>
<?php }else { ?>
<option value="<?php echo $this->escape($c->id_comercial); ?>"><?php
echo $this->escape($c->presentacion_comercial); ?></option>
<?php } ?>
<?php endforeach; ?>
</select
</td>
</tr>
</tbody>
</table>
<table style="text-align: left; width: 765px; vertical-align: middle;"
border="0">
<tbody>
```

```
<tr>
<td>
<label>STOCK MAXIMO:</label><label style="color: red">*</label>
</td>
<td>
<input name="stock_max" id="stock_max"  class="mayusculas"
size="20" type="text" onkeypress = "return validarNume(event)"/>
</td>
<td>
<label>STOCK MINIMO:</label><label style="color: red">*</label>
</td>
<td>
<input name="stock_min" id="stock_min"  class="mayusculas" size="20"
type="text" onkeypress = "return validarNume(event)"/>
</td>
</tr>
</tbody>
</table>
<br><br>
</fieldset>
<fieldset>
<br>
<table style="text-align: center; width: 765px; height: 52px;"
border="0" cellpadding="0" cellspacing="0">
<tbody>
<tr>
<td style="vertical-align: middle; width: 50%;">
<input onclick="sendform()" class="button small blue"
value="Registrar Producto" type="button"></td>
<td style="vertical-align: middle; width: 50%;">
        <input type="button" class="button small blue"
        onClick="document.location = '<?php echo $this-
        >url(array('controller' => 'producto',
          'action' => 'index'));?>'" name="Cancelar" value="Cancelar">
```

```
<br>
</td>
</tr>
</tbody>
</table>
<br>
</fieldset>
</form>
</div>
</center>
<BR>
<div id = "errores"></div>
```

De la misma manera suprimiremos todo lo relacionado con el botón "submit", para tener el control de cuándo se envían los datos del formulario al servidor con esto eliminaremos las validaciones que hacemos a nivel de cliente de nuestra clase Form, ya que esa función la controlamos desde nuestro javascript, una vez se hayan pasado todas las validaciones a nivel de cliente.

El método constructor de nuestra clase Producto, quedaría de la siguiente manera en el directorio: /public/js/validación_producto.js:

```
merror = "";

String.prototype.trim = function() {
return this.replace(/^\s+|\s+$/g, "");
};

function validarNume(e)
{
tecla = (document.all) ? e.keyCode : e.which;
if (tecla == 8 || e.keyCode == 9 ) return true;
patron = /\d/;
```

```
te = String.fromCharCode(tecla);
return patron.test(te);
}

function noNumero(valor){
   //Comprobamos si es número
if (/^([0-9])*$/.test(valor))
return false;
else
return true;
}

function validatefields(){
componentes = document.getElementById("producto").elements;
error = false;

for (i=0; i< componentes.length; i++){
switch (componentes[i].name){

case "stock_max" :if (componentes[i].value.trim().length == 0){
        merror = "El campo stock maximo es obligatorio";
        error = true;
                        document.producto.stock_max.focus()
                    }
        break;

case "stock_min" :if (componentes[i].value.trim().length == 0){
        merror = "El campo stock minimo es obligatorio";
        error = true;
                        document.producto.stock_min.focus()
                    }
        break;

case "observacion_producto" :if (componentes[i].value.trim().length ==
```

```
0){
        merror = "El campo CUM es obligatorio";
        error = true;
                        document.producto.observacion_producto.focus()
        }
        break;


case "nombre_producto" :if (componentes[i].value.trim().length == 0){
        merror = "El campo nombre producto es obligatorio";
        error = true;
                        document.producto.nombre_producto.focus()
                      }
        break;
       }
    }

        return (!error);
}

function eliminar_producto(id, producto){

var confir = confirm("¿Desea eliminar el registro del producto " + producto+" ?") ;
if(confir)
   {
       $.ajax(
     {
dataType: "html",
type: "POST",
url: "/sgmas-med/public/producto/delete", // ruta donde se encuentra nuestro action que procesa la peticion XmlHttpRequest
data: 'id_producto='+ id , //Se añade el parametro de busqueda por nombre
```

```
beforeSend: function(data){
            },
success: function(requestData){  //Llamada exitosa
alert("Registro eliminado correctamente");
              document.location = '/sgmas-med/public/producto';
            },
error: function(requestData, strError, strTipoError){
alert("Error " + strTipoError +': ' + strError ); //En caso de error mostraremos un alert
            },
complete: function(requestData){          //Llamada exitosa
            document.location = '/sgmas-med/public/producto';
            }
       });

   }
}

function actualizar_producto(){

if (validatefields()){
var id_producto = document.producto.id_producto.value;
var id_bodega =
document.producto.id_bodega.options[document.producto.id_bodega.selectedIndex].value;
var id_familia =
document.producto.id_familia.options[document.producto.id_familia.selectedIndex].value;
var id_presentacion =
document.producto.id_presentacion.options[document.producto.id_presentacion.selectedIndex].value;
var id_unidad =
document.producto.id_unidad.options[document.producto.id_unidad.selectedIndex].value;
```

```
var id_comercial =
document.producto.id_comercial.options[document.producto.id_comercia
l.selectedIndex].value;
var nombre_producto = document.producto.nombre_producto.value;
var estado_producto = document.producto.estado_producto.value;
var observacion_producto =
document.producto.observacion_producto.value;
var stock_max = document.producto.stock_max.value;
var stock_min = document.producto.stock_min.value;

//alert(id_producto+"-"+id_bodega+"-"+"-"+id_familia+"-"+"-
"+id_presentacion+"-"+"-"+id_unidad+"-"+"-"+nombre_producto);

$.ajax(
      {
dataType: "html",
type: "POST",
url: "/sgmas-med/public/producto/update", // ruta donde se encuentra
nuestro action que procesa la peticion XmlHttpRequest
          data: '&id_producto=' +id_producto + '&id_bodega='+ id_bodega
+ '&id_familia=' + id_familia + '&id_presentacion=' + id_presentacion +
'&id_unidad=' + id_unidad + '&id_comercial=' + id_comercial +
'&nombre_producto=' + nombre_producto + '&estado_producto=' +
estado_producto + '&observacion_producto=' + observacion_producto +
'&stock_max=' + stock_max + '&stock_min=' + stock_min, //Se añade el
parametro de busqueda ya sea por nombre o por especialidad
beforeSend: function(data){
             },
success: function(requestData){  //Llamada exitosa
alert("Datos actualizados correctamente");
           },
error: function(requestData, strError, strTipoError){
alert("Error " + strTipoError +': ' + strError ); //En caso de error
mostraremos un alert
```

```
            },
  complete: function(requestData){          //Llamada exitosa
              document.location = '/sgmas-med/public/producto';
  }
        });

  }else
    {
  alert(merror);
     }

  }

  function sendform(){

  if (validatefields()){
  alert("Registro guardado correctamente");
  document.getElementById("producto").submit();}
  else{
  alert(merror);

        }
  }
```

Cabe destacar que esta clase, aunque parezca que haya perdido su utilidad, no será así, ya que nos servirá para aplicar filtros o para hacer validaciones a nivel de servidor, pues así lograremos no hacer público el algoritmo de validación de cualquier dato.

La utilización de CSS en código HTML modificando las vistas de los formularios del sistema nos permite realizar formularios personalizados para dar estilos a las interfaces que se conviertan en una herramienta agradable a la vista del usuario y de fácil manipulación.

El archivo CSS se ubica en el directorio: /public/css/style.css que se invoca a la cabecera de cada archivo .phtm en la vista del formulario quedando de esta manera:

```
*
{
padding: 0;
margin: 0;
}

body
{
background-color: #E7E8E9;
background-image: url('../images/fondopage.jpg');
background-repeat: repeat;
background-position: left top;
font-family: Arial;
font-size: 12px;
padding: 0;
margin: 0;
color: #555;
line-height: 17px;
}

img
{
border-style: none;
}

a
{
color: #5C6F4C;
}
```

```
a:hover
{
text-decoration: none;
color: #999;
}

h3
{
border-bottom-color: #aaa;
border-bottom-width: 1px;
border-bottom-style: dotted;
color: #666;
padding-bottom: 4px;
margin-top: 0px;
margin-right: 0;
margin-bottom: 7px;
margin-left: 0;
font-weight: 100;
font-size: 22px;
letter-spacing: -1px;
}

h3 a
{
text-decoration: none;
font-size: 22px;
letter-spacing: -1px;
}

h3 a:hover
{
color: #999;
}
```

```
#wrap
{
width: 940px;
margin-top: 20px;
margin-right: auto;
margin-bottom: 20px;
margin-left: auto;
}

#header
{
background-color: #FFFFFF;
font-family: "Century Gothic", "Trebuchet MS", "Arial Narrow", Arial, sans-
serif;
height: 95px;
}

#header h1
{
font-size: 35px;
font-size: 28px;
font-weight: 100;
letter-spacing: -3px;
padding-top: 0;
padding-right: 0;
padding-bottom: 5px;
padding-left: 23px;

text-align: left;
}

#header h1 a
{
color: #5C6F4C;
```

```
text-decoration: none;
}

#header h1 a:hover
{
color: #111;
text-decoration: none;
}

#header h2
{
font-size: 13px;
color: #666;
padding-top: 4px;
padding-right: 0;
padding-bottom: 0;
padding-left: 23px;
text-align: left;
font-weight: 100;
}

#menu
{
background-image: url(../images/menu.png);
background-color: #6D6D6D;
background-repeat: no-repeat;
height: 32px;
line-height: 34px;
padding-left: 0px;
}

#menu li
{
float: left;
```

```
list-style-type: none;
}

#menu li a
{
display: block;
padding-top: 0;
padding-right: 10px;
padding-bottom: 0;
padding-left: 10px;
text-decoration: none;
color: #fff;
}

#menu li a:hover
{
color: #fff;
text-decoration: none;
background-image: url(../images/menuover.jpg);
background-color: #577141;
background-repeat: repeat-x;
}

#content
{
background-color: #FFFFFF;
padding-top: 10px;
padding-right: 10px;
padding-bottom: 10px;
padding-left: 10px;
}

#left
{
```

```
/* background-color: rgb(12, 33, 141);*/
background-color: #FFFFFF;
padding-top: 10px;
padding-right: 10px;
padding-bottom: 10px;
padding-left: 10px;
width: 210px;
height: 410px;
float: left;
font-size: 12px;
text-align: justify;
}

#left h3
{
/* color: rgb(76, 140, 206);*/
color: #555893;
font-size: 18px ;
border-bottom-color: rgb(76, 140, 206);
border-bottom-width: 1px;
border-bottom-style: dotted;
}

#right
{
/* background-color: rgb(203, 202, 255);*/
padding-top: 20px;
padding-right: 0;
padding-bottom: 20px;
padding-left: 30px;
width: 630px;
float: left;
text-align: justify;
}
```

```
#right h3
{
font-size: 17px;
border-style: none;
padding-top: 0;
padding-right: 0;
padding-bottom: 0;
padding-left: 10px;
margin: 0;
color: #111;
height: 30px;
line-height: 30px;
}

#right ul
{
list-style-type: none;
padding-top: 10px;
padding-right: 0;
padding-bottom: 20px;
padding-left: 20px;
}

#right ul li
{
padding-top: 2px;
padding-right: 0;
padding-bottom: 3px;
padding-left: 0;
}

#right ul li a
{
```

```
color: #5C6F4C;
font-weight: 100;
display: block;
text-decoration: none;
font-size: 14px;
}

#right ul li a:hover
{
color: #999;
}

#footer
{
background-image: url(../images/Footer.png);
background-color: #6D6D6D;
background-repeat: repeat;
font-size: 10px;
color: #16f;
text-align: center;
height: 38px;
}

#footer a
{
color: red;
text-decoration: none;
}

#footer a:hover
{
color: #aaa;
text-decoration: underline;
}
```

```
#logo
{
background-image: url("../images/logo.jpg");
background-repeat: no-repeat;
height: 95px;
width: 480px;
}

#logo2
{
background-image: url("../images/logo.jpg");
background-repeat: no-repeat;
height: 95px;
width: 480px;
}

#cabecera
{
background-image: url("../images/header.png");
background-repeat: no-repeat;
height: 40px;
left: 0;
top: 0;
width: 944px;
z-index: -2;
}

/*fieldset*/

fieldset{
font-family: sans-serif;
color: #555893;
font-weight: bold;
```

```
font-size: 10pt;
 }


 /*formularios*/
form label{
font-family: sans-serif;
color: #0B0B61;
font-weight: bold;
font-size: 8pt;
 }
 /*ingresar solo mayusculas en el form*/
input.mayusculas{text-transform:uppercase;}
/*botones*/
a.button2{
background:url("../images/botones/boton.png");
display:block;
color: #0B0B61;
font-weight:bold;
height:30px;
line-height:29px;
margin-bottom:14px;
text-decoration:none;
width:191px;
}
a:hover.button2{
color:#0066CC;
}
/* -------------------- */
/* CLASSES */
/* -------------------- */
.add{
background:url("../images/botones/add.ico") no-repeat 10px 8px;
text-indent:30px;
```

```
display:block;
}
.delete{
background:url("../images/botones/delete.ico") no-repeat 10px 8px;
text-indent:30px;
display:block;
}
.cancel{
background:url("../images/botones/cancel.ico") no-repeat 10px 8px;
text-indent:30px;
display:block;
}
.update{
background:url("../images/botones/update.ico") no-repeat 10px 8px;
text-indent:30px;
display:block;
}
```

Con esto tenemos un mayor control de los estilos de todos los componentes del formulario dando una imagen personalizada y agradable para el usuario acorde a la plantilla que se utiliza para el proyecto.

4.3 CONEXIÓN CON BASE DE DATOS

Ahora podemos definir las reglas de negocio en las que está basada nuestra aplicación. Estas reglas de negocio son lo que constituyen el Modelo, y están basadas en los datos que se encuentran en nuestra base de datos.

La base de datos utilizada está íntegramente realizada en Postgres teniendo acceso a nuestra base de datos de la aplicación a través del archivo de configuración application.ini, siendo la siguiente:

```
resources.db.adapter = PDO_PGSQL
resources.db.params.host = localhost
resources.db.params.username = majo
resources.db.params.password = admin
resources.db.params.dbname = sgmas
```

Zend Framework facilita la clase Zend_Db_Table la cual permite acceder a los datos que se encuentra en la Base de Datos. Para proyectos de mayor envergadura, se suelen crear una o más instancias de la clase Zend_Db_Table.

Zend_Db_Table es una clase abstracta, por lo que tenemos que crear una que herede de ella. El nombre que tendrá, vendrá definida por el prefijo Model_DbTable_ y el nombre de la tabla con la que se va a interactuar, nombre que además informaremos en la variable protegida $_name.

Zend_Db_Table asume que nuestra tabla tiene una clave primaria que se llama id y que este es un auto incremental. El nombre de la clave primaria se puede modificar si así se desea.

Nuestra nueva clase la crearemos en el archivo Producto.php, dentro de /applications/models/DbTable/producto.php, desde este archivo podemos controlar todos los datos guardados en la base en forma de métodos que mediante instancias podemos acceder a las funciones que se encuentran en el mismo. De esta forma tenemos conformado el archivo Producto.php que maneja el modelo de la base de datos:

```
<?php

class Application_Model_DbTable_Producto extends
Zend_Db_Table_Abstract
{

protected $_name = 'tb_producto';
```

```
public function get($id_producto)
    {
        $id_producto = (int) $id_producto;
        //$this->fetchRow devuelve fila donde id = $id
        $row = $this->fetchRow('id_producto = ' . $id_producto);
if (!$row)
        {
throw new Exception("No se puede encontrar el registro
$id_producto");
        }
return $row->toArray();
    }

public function get1($id_producto)
    {
        $id_producto = (int) $id_producto;
        //$this->fetchRow devuelve fila donde id = $id
return $this->fetchRow('id_producto = ' . $id_producto);
    }



    /**
     *  agrega un nuevo producto a la base de datos
*/
public function agregar($id_bodega, $id_familia, $id_presentacion,
$id_unidad, $id_comercial ,$nombre_producto, $estado_producto,
$observacion_producto, $stock_max, $stock_min)
{
        $data = array('id_bodega' => $id_bodega, 'id_familia' =>
$id_familia, 'id_presentacion'=> $id_presentacion,
        'id_unidad' => $id_unidad, 'id_comercial' => $id_comercial,
'nombre_producto' => $nombre_producto, 'estado_producto' =>
$estado_producto,
```

```
'observacion_producto' => $observacion_producto, 'stock_max' =>
$stock_max, 'stock_min' => $stock_min);
        //$this->insert inserta nuevo album
        $this->insert($data);
}

public function cambiar($id_producto,$id_bodega, $id_familia,
  $id_presentacion, $id_unidad, $id_comercial,$nombre_producto,
  $estado_producto, $observacion_producto, $stock_max, $stock_min)
    {
$data = array('id_producto' => $id_producto, 'id_bodega' =>
$id_bodega, 'id_familia' => $id_familia, 'id_presentacion'=>
    $id_presentacion, 'id_unidad' => $id_unidad, 'id_comercial' =>
    $id_comercial, 'nombre_producto' => $nombre_producto,
    'estado_producto' => $estado_producto,
        'observacion_producto' => $observacion_producto, 'stock_max' =>
$stock_max, 'stock_min' => $stock_min);
//$this->update cambia datos del producto con id= $id
$this->update($data, 'id_producto = ' . (int) $id_producto);
    }

public function borrar($id)
{
//$this->delete borrar album donde id=$id
$this->delete('id_producto =' . (int) $id);
    }

public function listar()
{
//devuelve todos los registros de la tabla
return $this->fetchAll();
    }

public function listar_reporte()
```

```
    {

        $db = Zend_Registry::get('pgdb');
        $db->setFetchMode(Zend_Db::FETCH_OBJ);
//hago la consulta sql que desee:
        //
         //devuelve todos los registros de la tabla
return $db->fetchAll("SELECT P.ID_PRODUCTO,
B.NOMBRE_BODEGA, F.NOMBRE_FAMILIA,
       PR.NOMBRE_PRESENTACION, U.UNIDAD_MEDIDA,
         P.NOMBRE_PRODUCTO, P.ESTADO_PRODUCTO,
P.OBSERVACION_PRODUCTO, P.STOCK_MAX, P.STOCK_MIN
       FROM TB_PRODUCTO P, TB_BODEGA B, TB_FAMILIA F,
TB_PRESENTACION PR, TB_UNIDADES U
WHERE P.ID_BODEGA=B.ID_BODEGA AND
      P.ID_FAMILIA=F.ID_FAMILIA AND
P.ID_PRESENTACION=PR.ID_PRESENTACION AND
      P.ID_UNIDAD=U.ID_UNIDAD");
    }

public function buscar_producto1($consulta)
    {
        //recupero objeto desde el registro
$db = Zend_Registry::get('pgdb');
//opcional, esto es para que devuelva los resultados como objetos $row-
>campo
$db->setFetchMode(Zend_Db::FETCH_OBJ);

return $db->fetchRow("SELECT ESTADO_PRODUCTO
                        FROM TB_PRODUCTO
                        WHERE          NOMBRE_PRODUCTO          LIKE
'%".$consulta."%'");
    }
```

```
public function consulta_produc($consulta)
    {

        $db = Zend_Registry::get('pgdb');
//opcional, esto es para que devuelva los resultados como objetos $row->campo
$db->setFetchMode(Zend_Db::FETCH_OBJ);
//hago la consulta sql que desee:
$select="SELECT *
                FROM tb_producto AS p
                INNER JOIN tb_bodega AS b ON (p.id_bodega = b.id_bodega)
                INNER JOIN tb_familia AS f ON (p.id_familia = f.id_familia)
                INNER JOIN tb_presentacion AS pr ON (p.id_presentacion = pr.id_presentacion)
INNER JOIN tb_unidades AS u ON (p.id_unidad = u.id_unidad)
WHERE (p.nombre_producto LIKE '%".$consulta."%')
                OR (p.observacion_producto LIKE '%".$consulta."%')";

return $db->fetchAll($select);
    }

public function consulta_producto($parametro)
    {

        $db = Zend_Registry::get('pgdb');
//opcional, esto es para que devuelva los resultados como objetos $row->campo
$db->setFetchMode(Zend_Db::FETCH_OBJ);
//hago la consulta sql que desee:
return $db->fetchAll("SELECT * FROM TB_PRODUCTO AS p
                        INNER JOIN tb_presentacion AS pr ON (p.id_presentacion = pr.id_presentacion)
INNER JOIN tb_unidades AS u ON (p.id_unidad = u.id_unidad)
```

```
                WHERE (p.NOMBRE_PRODUCTO LIKE
'%".$parametro."%')
                    OR (p.OBSERVACION_PRODUCTO LIKE
'%".$parametro."%')");
    }
}
```

4.4 CREACIÓN DE PLANTILLA PRINCIPAL

El componente de Zend Framework que da soporte a las vistas es Zend_View. Este nos permite separar la presentación de la lógica o reglas de negocio que se encuentra en el modelo.

En Zend Framework encontramos las vistas en el directorio “views” y se organizan de la forma:

views/scripts/{nombredelcontrolador}/{nombredelaaccion}.phtml.

En la mayoría de los proyectos hay partes de código HTML que se repite para todas las vistas, por ejemplo: un encabezado, una columna lateral y el pie de página. Para evitar repetir código es que existe la posibilidad de crear un “layout” o plantilla donde colocaremos el código común y desde donde llamaremos las vistas.

Lo primero que debemos hacer es crear el directorio: “application/layouts/” y agregar en nuestro archivo de configuración la siguiente el siguiente código HTML que corresponde a la plantilla creada para la realización del proyecto y acorde a las exigencias del Centro de Salud La Tola – Vicentina y el Ministerio Salud Pública para ello modificamos el archivo application.ini el la siguiente línea:

```
resources.layout.layoutpath = APPLICATION_PATH “/layouts”
```

En la cual colocamos dentro de la sección [production] del archivo de configuración de la aplicación.

Además, necesitamos inicializar nuestra vista en la clase Boostrap, para lo que crearemos otros métodos que comiencen por el prefijo _init justo debajo de _initAutoload(). El método se llamará _initViewHelpers()

```
<?php

class Bootstrap extends Zend_Application_Bootstrap_Bootstrap
{

protected function _initAutoload(){
    $moduleLoader = new
Zend_Application_Module_Autoloader(array('namespace'        =>        '',
'basePath' => PLICATION_PATH));
return $moduleLoader;
  }

protected function _initView()
  {
    $this->bootstrap('layout');
    $layout = $this->getResource('layout');
    $view = $layout->getView();
    $view = new Zend_View();
    $view->setEncoding('UTF-8');
    $view->doctype('XHTML1_STRICT');
    $view->headMeta()->appendHttpEquiv('Content-Type',
'text/html;charset=utf-8');

//Agrego el path de ZendX para poder usar JQuerry
$view->headTitle()->setSeparator(' - ');
    $view->headTitle('Gestion de Medicamentos');
```

```
    $view->addHelperPath('ZendX/JQuery/View/Helper/',
'ZendX_JQuery_View_Helper');
    $viewRenderer                                                    =
Zend_Controller_Action_HelperBroker::getStaticHelper('ViewRenderer');
    $viewRenderer->setView($view);
return $view;
  }
protected function _initViewHelpers()
  {
    $this->bootstrap('layout');
    $layout = $this->getResource('layout');
    $view = $layout->getView();
    $view = new Zend_View();
    $view->setEncoding('UTF-8');
    $view->doctype('XHTML1_STRICT');
    $view->headMeta()->appendHttpEquiv(
                    'Content-Type', 'text/html;charset=utf-8'
    );
    $view->headTitle()->setSeparator(' - ');
    $view->headTitle('Gestion de Medicamentos');

    $viewRenderer                                                    =
Zend_Controller_Action_HelperBroker::getStaticHelper(
                'ViewRenderer'
    );
    $viewRenderer->setView($view);

    Zend_Session::start();
return $view;

  }
}
```

A través del método bootstrap() inicializamos la vista para cuando después recuperemos el objeto Zend_Layout a través de getResource(), para después recuperar la vista utilizando el método getView().

Una vez que tenemos la vista instanciada, podemos llamar a varios métodos de ayuda de la vista, que definirán en nuestra vista el código correspondiente para cada uno de ellos.

El código que tendremos en nuestra vista “layout.phtml” y que tendremos en el directorio /application/layouts sería el siguiente para todos nuestros formularios que deseemos aplicar la plantilla creada:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<!--<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />-->
<?php echo $this->headMeta(); ?>
<?php echo $this->headTitle(); ?>

<?php echo $this->headScript(); ?>
<?php echo $this->headLink(); ?>
<!-- Menu plantilla-->
<link href="<?php echo $this->baseUrl; ?>/sgmas-med/public/css/style.css" media="screen" rel="stylesheet" type="text/css" />
<link href="<?php echo $this->baseUrl; ?>/sgmas-med/public/css/dropdown/dropdown.css" media="screen" rel="stylesheet"
type="text/css" />
<link href="<?php echo $this->baseUrl; ?>/sgmas-med/public/css/dropdown/themes/nvidia.com/default.advanced.css"
media="screen" rel="stylesheet" type="text/css" />
```

```
<link href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/images/favicon(1).ico" type="image/x-icon" rel="shortcut icon"
/>
<link href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/js/jquery/jquery-
1.4.3.js" type="image/x-icon" rel="shortcut icon" />
<!--<link rel="icon" type="image/png"
href="images/themes/default/Introduction.png" />-->
<!--[if lt IE 7]>
<script type="text/javascript" src="js/jquery/jquery.js"></script>
<script type="text/javascript" src="js/jquery/jquery.dropdown.js"></script>
<![endif]-->

</head>
<body>
<div id="wrap">
<div id="cabecera">
<?php if($this->user) : ?>
<p style="color: white; font-size: 10.5px" id="logged-in"
align='right'><?php
echo $this->escape($this->user->perfil);?> |  
<img src="<?php echo $this->baseUrl; ?>/images/logout.png"
alt="cerrar sesion" ></img>
<a style="color: white;" href="<?php echo $this->baseUrl
?>/index/logout">cerrar sesiÃ³n</a>
      
</p>
<?php endif; ?>
</div>
<div id="header">
        <table>
        <tr>
<td><div id="logo"></div></td>
<td align = "right" ><h1>Area de Salud Nro. 3</h1><h2>La Tola –
```

```
Vicentina</h2></td>
        </tr>
        </table>
</div>

<div id="menu">
<ul class="dropdown dropdown-horizontal" >
        <li><a href="./" class="dir">REGISTROS DE
MEDICAMENTOS</a>
                <ul>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/producto">Productos</a></li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/ubicacion">Ubicaci&oacute;n F&iacute;sica</a></li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/bajas">Registro de Bajas</a></li>

<!--<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/lote">Lotes de medicaci&oacute;n</a></li>-->
</ul>
        </li>
        <li><a href="./" class="dir">MANTENIMIENTO</a>
                <ul>
<li><a href="./">PRODUCTOS</a></li>
<li><a href="./"></a>.....</li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/bodega">Bodega</a></li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/familia">Familia</a></li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/presentacion">Presentaci&oacute;n</a></li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/responsable">Responsables</a></li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
```

```
med/public/unidades">Unidades</a></li>
<li><a href="./"></a>.....</li>
<li><a href="./">LOTE</a></li>
<li><a href="./"></a>.....</li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/proveedores">Proveedores</a></li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/tipo">Documento Contable</a></li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/documento">Registro Documento Contable</a></li>
</ul>
        </li>
<li><a href="<?php echo $this->baseUrl; ?>/sgmas-med/public/lote"
class="dir">INGRESOS</a>
</li>
        <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/despachos"
class="dir">EGRESOS</a>
<!--            <ul>
                    <li><a href="<?php //echo $this->baseUrl; ?>/sgmas-
med/public/despachos">Despachos</a></li>
                    <li><a href="./">Opcion 4C</a></li>
                    <li><a href="./">Opcion 4D</a></li>
                    <li><a href="./">Opcion 4E</a></li>
                    <li><a href="./">Opcion 4F</a></li>
                    <li><a href="./">Opcion 4G</a></li>
                    <li><a href="./">Opcion 4H</a></li>
            </ul>-->
        </li>
        <li><a href="./" class="dir">REPORTES</a>
            <ul>
                    <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/cantidad">Stock lotes</a></li>
                    <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
```

```
med/public/preciounitario">Precio Unitario Promedio</a></li>
                    <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/despachos/imprimirdespachos">Imprimir Despachos</a></li>
                    <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/reportedespachosperiodos/">Reporte Periodos
Despachos</a></li>
                    <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/registro-contable/">Reporte Registro Contable</a></li>
                    <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/control-ingresos/">Control Ingresos</a></li>
<!--                <li><a href="./"></a></li>
                    <li><a href="./"></a></li>-->
                </ul>
        </li>
        <li><a href="./" class="dir">USUARIOS</a>
<!--            <ul>
                    <li><a href="./">Opcion 6A</a></li>
                    <li><a href="./">Opcion 6B</a></li>
                    <li><a href="./">Opcion 6C</a></li>
                    <li><a href="./">Opcion 6D</a></li>
                    <li><a href="./">Opcion 6E</a></li>
                    <li><a href="./">Opcion 6F</a></li>
                    <li><a href="./">Opcion 6G</a></li>
                    <li><a href="./">Opcion 6H</a></li>
                    <li><a href="./">Opcion 6I</a></li>
                </ul>-->
        </li>
</ul>

<!-- / END -->

<!--    <li><a href="#">Page1</a></li> -->
<!--    <li><a href="#">Page2</a></li> -->
<!--    <li><a href="#">Page3</a></li> -->
```

```
<!--     <li><a href="#">Page4</a></li> -->
</div>
<div id="content">
<center>
<?php echo $this->escape($this->title); ?>
<?php echo $this->layout()->content; ?>
</center>
<div style="clear:both;">
</div>
</div>
        <div id="footer">
                <h2>UNIVERSIDAD POLITECNICA SALESIANA</h2>
                <h2>Ingenieria en Sistemas - Telematica / Robotica </h2>
<h2><a href="<?php echo $this->baseUrl ?>/sgmas-
med/public/auth/logout">Salir del Sistema</a></h2>
        </div>
</div>
</body>
</html>
```

En el código de nuestra página, vemos como tenemos una instancia del objeto de la vista a través de $this, por lo que podemos utilizar esta instancia para llamar a los métodos asociados a la vista y recuperar información que le ha sido asignada a la vista.

4.5 PRUEBAS

4.5.1 PRUEBA DE CAJA NEGRA

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (en sentido general: teclado, pantalla, ficheros, canales de comunicaciones, etc.).

Las pruebas de caja negra se apoyan en la especificación de requisitos del módulo. De hecho, se habla de "cobertura de especificación" para dar una medida del número de requisitos que se han probado. Es fácil obtener coberturas del 100% en módulos internos, aunque puede ser más laborioso en módulos con interfaz al exterior. En cualquier caso, es muy recomendable conseguir una alta cobertura en esta línea.

El problema con las pruebas de caja negra no suele estar en el número de funciones proporcionadas por el módulo (que siempre es un número muy limitado en diseños razonables); sino en los datos que se le pasan a estas funciones. El conjunto de datos posibles suele ser muy amplio (por ejemplo, un entero).

A la vista de los requisitos de un módulo, se sigue una técnica algebraica conocida como "clases de equivalencia". Esta técnica trata cada parámetro como un modelo algebraico donde unos datos son equivalentes a otros. Si logramos partir un rango excesivamente

amplio de posibles valores reales a un conjunto reducido de clases de equivalencia, entonces es suficiente probar un caso de cada clase, pues los demás datos de la misma clase son equivalentes.

Durante la lectura de los requisitos del sistema, nos encontraremos con una serie de valores singulares, que marcan diferencias de comportamiento. Estos valores son claros candidatos a marcar clases de equivalencia: por abajo y por arriba.

Una vez identificadas las clases de equivalencia significativas en nuestro módulo, se procede a coger un valor de cada clase, que no esté justamente al límite de la clase. Este valor aleatorio, hará las veces de cualquier valor normal que se le pueda pasar en la ejecución real.

La experiencia muestra que un buen número de errores aparecen en torno a los puntos de cambio de clase de equivalencia. Hay una serie de valores denominados "frontera" (o valores límite) que conviene probar, además de los elegidos en el párrafo anterior. Usualmente se necesitan 2 valores por frontera, uno justo abajo y otro justo encima.

4.5.1.1 Caso de prueba

4.5.1.1.1 Logeado de Usuario para Acceso al Sistema

ACCIONES	LOGEADO DE USUARIO PARA ACCESO AL SISTEMA
Objetivos	Verificar el logueado de los usuarios con los parámetros de login y password para el acceso del sistema.
Observaciones	Cabe destacar que el proyecto en general tiene su propio módulo de administración de usuarios y que el módulo de Gestión de medicamentos funciona para la presentación de su eficaz funcionamiento con un sistema de gestión de usuarios propio pero fácilmente acoplable por su arquitectura, diseño y construcción del mismo.
Descripción	El sistema debe verificar e identificar eficiente el logueado de los usuarios del sistema y permitir el ingreso a las diferentes interfaces del mismo.
Proceso	Acción 1 Usuario ejecuta el inicio para carga del sistema 2 Sistema solicita autenticación del usuario 3 El usuario ingresa su login y password asignados para ingreso al sistema 4 Verificar el ingreso del login del usuario que se lo realiza mediante el número de cédula del mismo,

		mientras que el password dependiendo del login ingresado procede a verificar su código de ingreso asignado para el mismo.
	5	Validar parámetros ingresados por los usuarios del sistema.
	6	INGRESAR al sistema
	7	Verificar los parámetros ingresados por el usuarios para el ingreso al sistema dependiendo del perfil que posea (Administrador y superadministrador).
		Acción
	1	El IIS, mozilla, opera o cualquier browser que se utilice debe estar correctamente instalado y/o actualizado no se generan errores en cuanto al funcionamiento del sistema sino que existen pequeñas inconsistencias al momento de mostrar las diferentes interfaces del sistema y las plantillas del mismo.
	2	Al ser un sistema desarrollado en software libre y probado para el efecto en mozilla se recomienda el uso de ese browser para su eficaz y correcto funcionamiento.
Excepciones		
	3	Si el equipo o servidor donde se aloje el sistema no cumple los requisitos mínimos para el funcionamiento del sistema.
	4	Si los parámetros de login y password no son ingresados correctamente el sistema solicita redefinir los parámetros antes ingresados nuevamente mediante un menaje de error de usuario o password

incorrectos.

5 La validación de los datos de usuario se los hace mediante un script de verificación de datos únicamente numéricos y con el tamaño específico que requiere la cédula de identidad del usuario.

Tabla: 4.5.1.1.1 Logeado de Usuario para Acceso al Sistema.[81]

4.5.1.1.1.1 Prueba de Unidad de Caja Negra

ACCIONES	PRUEBAS DE CAJA NEGRA			
Caso de Prueba	Logueado de usuario para acceso al sistema			
Objetivos	Controlar que el logueado de usuario tiene control de duplicidad en los parámetros de login y password para el acceso al mismo dependiendo del perfil que el usuario posea.			
Resultado esperado	Denegar el acceso al sistema si los parámetros ingresados por el usuario son incorrectos.			
Condiciones de	Valores			
	Entradas	Tipo	Válido	No válido
	1 Usuario	Parámetro	Tipo numérico del tamaño del número de	Tipo carácter diferente al numérico.

[81] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Entrada				la cédula del usuario.	
	2	Password	Parámetro	Tipo alfanumérico de tamaño 50 con una encriptación MD5.	Tipo alfanumérico con una longitud mayor a la determinada.
		Condiciones			
	1	Cuando existe el parámetro de usuario con su respectivo número de identificación pero su password es incorrecto o existe en la tabla de usuarios en la base de datos.			
Condiciones de Ejecución	2	Cuando no existe el parámetro de usuario y su password es incorrecto en la tabla de usuarios en la base de datos.			
	3	Cuando el usuario es correcto y su password también lo es pero no se ha asignado ningún tipo de usuario para ingreso al sistema			

Tabla: 4.5.1.1.1.1 Prueba de Unidad de Caja Negra[82]

4.5.1.1.1.2 Pruebas de Unidad de caja Negra con Parámetros Ingresados

ACCIONES	PRUEBAS DE CAJA NEGRA CON VALORES

[82] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Parámetros		Valores	Resultados Esperados	Resultados Reales	Resultados Obtenidos
1	Usuario	17163859	Despliega un mensaje de error y prohíbe el ingreso al sistema.	Despliega un mensaje de error y prohíbe el ingreso al sistema.	OK
	Password	carlos	Despliega un mensaje de error y prohíbe el ingreso al sistema.	Despliega un mensaje de error y prohíbe el ingreso al sistema.	OK
2	Usuario	1716385925	Validación correcta y permiso adquirido para ingreso al sistema.	Validación correcta y permiso adquirido para ingreso al sistema.	OK
	Password	carlos1037	Validación correcta y permiso adquirido para ingreso al sistema.	Validación correcta y permiso adquirido para ingreso al sistema.	OK

Tabla: 4.5.1.1.1.2 Pruebas de Unidad de Caja Negra con Parámetros Ingresados[83]

4.5.1.1.1.3 Resultados obtenidos en Pruebas de Caja Negra

RESULTADOS OBTENIDOS EN PRUEBAS DE CAJA NEGRA			
Fecha	Hora inicio	Hora fin	Observaciones
2011-11-03	8:30 am	10:00 am	La gestión de logeado de usuario y contraseña se la pudo validar y verificar de manera exitosa logrando los resultados esperados para las pruebas pertinentes.

Tabla: 4.5.1.1.1.3 Resultados obtenidos en Pruebas de Caja Negra[84]

4.5.1.1.2 Ingreso de un Lote de Medicación al Sistema

ACCIONES	INGRESO DE UN LOTE DE MEDICACIÓN AL SISTEMA
Objetivos	Verificar el correcto ingreso de un lote de medicación al sistema.
Descripción	El sistema debe poder ingresar, editar lotes de medicación que ingresan a bodega para su posterior verificación de

[83] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*
[84] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

existencias y despacho del mismo.

	Acción
1	Usuario inicia interfaz de INGRESOS en el menú principal del sistema.
2	Verificar la existencia del producto y/o medicamento para su ingreso.
3	Determinar los valores faltantes en el formulario para guardar correctamente el medicamento ingresado.
4	Verificar y comprobar los datos que se han ingresado por medio del sistema. Cada uno de los datos están validados mediante un script que determina si el campo ingresado cumple con los parámetros determinados para su validación.
5	Determinar valores de numéricos importantes en el ingreso como son costo total del lote de medicación, su stock máximo y mínimo para control del mismo.
6	INGRESAR nuevos valores de lotes de medicación.
7	Verificar los datos correctamente ingresados en la pantalla principal de ingresos de lotes de medicación mediante filtros de búsquedas y visualización de datos indexados.

Proceso

	Acción
1	No existencia de datos importantes para el ingreso de un nuevo lote de medicación como por ejemplo la no existencia de un producto y/o medicamento para el ingreso de los lotes de medicación.

Excepciones	2	No existencia de datos secundarios pero necesarios para el ingreso de un nuevo lote de medicación en el caso de no existir se debe ingresar o al momento de olvidarse de marcar uno de esos campos se muestra un error mediante la validación de datos por medio de un script que se encarga de la validación del mismo.
	3	Inconsistencias en el código de ingreso de los datos mediante el controlador que los verifica.
	4	Mal funcionamiento del motor de base de datos POSTGRESQL el cual de presentarse el problema se despliega un mensaje de error y se procede a cancelar el guardado de los datos.
	5	Inconsistencias al ingreso de los datos que se manejan mediante los métodos que controlan la acción de guardado de datos.

Tabla: 4.5.1.1.2 Ingreso de un Lote de Medicación al Sistema[85]

4.5.1.1.2.1 Prueba de Unidad de Caja Negra

ACCIONES	PRUEBAS DE CAJA NEGRA
Caso de Prueba	Verificar el correcto ingreso de un lote de medicación al sistema.
Objetivos	Controlar que el ingreso de un nuevo lote de medicación funcione correctamente con los parámetros validados y verificados correctamente.
Resultado esperado	Ingresar un nuevo lote de medicación sin tener ningún inconveniente al verificar los datos y la verificación de las

[85] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

validaciones de los mismos.

Valores

		Entradas	Tipo	Válido	No válido
Condiciones de Entrada	1	No. Lote	Parámetro	Tipo alfanumérico.	Tipo alfanumérico con una longitud mayor a la determinada.
	2	Producto	Parámetro	Tipo alfanumérico que se carga según resultados de búsqueda.	No existencia del producto indicado.
	3	Nombre del producto	Parámetro	Tipo alfanumérico que se carga según resultados de búsqueda.	No existencia del producto indicado.
	4	Documento No. (Número del documento contable)	Parámetro	Tipo alfanumérico que se carga según resultados de búsqueda.	No existencia del producto indicado.
	5	Responsable	Parámetro	Tipo alfanumérico	Tipo alfanumérico

	del ingreso		ingresado por usuario.	ingresado por usuario sea de longitud mayor a la determinada.
6	Transacción	Parámetro	Tipo alfanumérico que se carga según resultados de búsqueda.	No existencia del producto indicado.
7	Programa	Parámetro	Tipo alfanumérico que se carga según resultados de búsqueda.	No existencia del producto indicado.
8	Fecha de ingreso	Parámetro	Tipo date que es igual a la fecha del sistema y se carga al momento del ingreso a la interfaz.	Datos del sistema como la fecha se encuentren erróneos o tipo de dato ingresado sea incorrecto en formato.
9	Fecha de	Parámetro	Tipo date	Tipo de dato

	caducidad		con formato correcto AAAA-mm-dd.	ingresado sea incorrecto en formato.
10	Fecha de elaboración	Parámetro	Tipo date con formato correcto AAAA-mm-dd.	Tipo de dato ingresado sea incorrecto en formato.
11	Costo lote	Parámetro	Tipo numérico de valor flotante para indicar el valor de la factura.	Tipo de dato que sobrepase el límite del dato ingresado.
12	Cantidad Lote	Parámetro	Tipo numérico que indica el stock indicado.	Tipo de dato diferente al ingresado.
13	Observaciones	Parámetro	Tipo alfanumérico de tamaño 50 de longitud.	Excederse en la longitud del campo para el ingreso de alguna observación.

Tabla: 4.5.1.1.2.1 Prueba de Unidad de Caja Negra[86]

[86] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

4.5.1.1.2.2 Pruebas de Unidad de Caja Negra con Parámetros Ingresados

ACCIONES		PRUEBAS DE CAJA NEGRA CON VALORES		
Parámetros	Valores	Resultados Esperados	Resultados Reales	Resultados Obtenidos
No. Lote	AAC-09333EW222	Despliega un mensaje de error que el dato proporcionado es incorrecto.	Despliega un mensaje de error que el dato proporcionado es incorrecto.	OK
Producto	EDC-001	Despliega un mensaje de error que el dato proporcionado es incorrecto o búsqueda infructuosa.	Despliega un mensaje de error que el dato proporcionado es incorrecto o búsqueda infructuosa.	OK
Nombre del producto	Amoxicilin	Despliega un mensaje de error que el dato proporcionado es incorrecto o búsqueda	Despliega un mensaje de error que el dato proporcionado es incorrecto o búsqueda	

1

		infructuosa.	infructuosa.	OK
Documento No. (Número del documento contable)	001_001_022	Despliega un mensaje de error que el dato proporcionado es incorrecto o búsqueda infructuosa.	Despliega un mensaje de error que el dato proporcionado es incorrecto o búsqueda infructuosa.	OK
Responsable del ingreso	Carlos Arias	N/I	N/I	OK
Transacción	Ingreso	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se procede a cargar en el combobox de la base de datos.	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se procede a cargar en el combobox de la base de datos.	OK

Programa	Cáncer	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se procede a cargar en el combobox de la base de datos.	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se procede a cargar en el combobox de la base de datos.	OK
Fecha de ingreso	11-23-2012	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se ajusta al formato determinado para ese tipo de dato AAAA-mm-dd.	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se ajusta al formato determinado para ese tipo de dato AAAA-mm-dd.	OK
Fecha de caducidad	11_12_2001	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se ajusta	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se ajusta al formato determinado para ese	

		al formato determinado para ese tipo de dato AAAA-mm-dd.	tipo de dato AAAA-mm-dd.	
				OK
Fecha de elaboración	11_22_2005	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se ajusta al formato determinado para ese tipo de dato AAAA-mm-dd.	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se ajusta al formato determinado para ese tipo de dato AAAA-mm-dd.	OK
Costo lote	2000.23	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se ajusta al formato determinado para ese tipo de dato	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se ajusta al formato determinado para ese tipo de dato flotante.	

		flotante.		OK
Cantidad Lote	1000,23	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se ajusta al formato determinado para ese tipo de dato.	Despliega un mensaje de error que el dato proporcionado es incorrecto o no se ajusta al formato determinado para ese tipo de dato.	OK
Observaciones	Cualquier tipo de comentario que posea el lote del producto se vaya a ingresar.	Despliega un mensaje de error que el dato proporcionado es incorrecto o excede el límite de longitud para ese campo.	Despliega un mensaje de error que el dato proporcionado es incorrecto o excede el límite de longitud para ese campo.	OK
No. Lote	A001-A005	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	OK

	Producto	1	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	
					OK
	Nombre del producto	Amoxicilina	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	
					OK
2	Documento No. (Número del documento contable)	001-001-001	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	
					OK
	Responsable del ingreso	Carlos Arias	Validación correcta e ingreso al sistema del	Validación correcta e ingreso al sistema del dato a la base.	

		dato a la base.		
				OK
Transacción	Ingreso a bodega	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	
				OK
Programa	Fiscal / Maternidad Gratuita	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	
				OK
Fecha de ingreso	2012-07-09	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	

				OK
Fecha de caducidad	2012-07-09	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	
				OK
Fecha de elaboración	2012-07-09	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	
				OK
Costo lote	2000,23	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	

				OK
Cantidad Lote	1000	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	
				OK
Observaciones	Cualquier tipo de comentario que posea el lote del producto se vaya a ingresar.	Validación correcta e ingreso al sistema del dato a la base.	Validación correcta e ingreso al sistema del dato a la base.	
				OK

Tabla: 4.5.1.1.2.2 Pruebas de Unidad de Caja Negra con Parámetros Ingresados[87]

[87] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

4.5.1.1.2.3 Resultados obtenidos en Pruebas de Caja Negra

RESULTADOS OBTENIDOS EN PRUEBAS DE CAJA NEGRA

Fecha	Hora inicio	Hora fin	Observaciones
2011-11-03	10:30 am	12:30 pm	La gestión de ingreso de nuevos lotes de medicación se realizó satisfactoriamente logrando los resultados esperados para las pruebas pertinentes.

Tabla: 4.5.1.1.2.3 Resultados obtenidos en pruebas de caja negra[88]

4.5.2 PRUEBA DE CAJA BLANCA

Consiste en realizar pruebas para verificar que líneas específicas de código funcionan tal como está definido. También se le conoce como prueba de caja-transparente.

Las pruebas de caja blanca intentan garantizar que:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecuten todos los bucles en sus límites.
- Se utilizan todas las estructuras de datos internas.

[88] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Para esta prueba se consideran tres importantes puntos:

- Conocer el desarrollo interno del programa, determinante en el análisis de coherencia y consistencia del código.
- Considerar las reglas predefinidas por cada algoritmo.
- Comparar el desarrollo del programa en su código con la documentación pertinente.

Como el ordenador está obligado a ejecutarlas una tras otra, es lo mismo decir que se han ejecutado todas las sentencias o todos los segmentos.

El número de sentencias de un programa es finito. Basta coger el código fuente e ir contando. Se puede diseñar un plan de pruebas que vaya ejecutando más y más sentencias, hasta que hayamos pasado por todas, o por una inmensa mayoría.

En la práctica, el proceso de pruebas termina antes de llegar al 100%, pues puede ser excesivamente laborioso y costoso provocar el paso por todas y cada una de las sentencias.

A la hora de decidir el punto de corte antes de llegar al 100% de cobertura hay que ser precavido y tomar en consideración algo más que el índice conseguido. En efecto, ocurre con frecuencia que los programas contienen código muerto o inalcanzable. Puede ser que este trozo del programa, simplemente "sobre" y se pueda prescindir de él; pero a veces significa que una cierta funcionalidad, necesaria, es inalcanzable: esto es un error y hay que corregirlo.

Desde el punto de vista de cobertura de segmentos, basta ejecutar una vez, con éxito en la condición, para cubrir todas las sentencias posibles. Sin embargo, desde el punto de vista de la lógica del

programa, también debe ser importante el caso de que la condición falle (si no lo fuera, sobra el IF). Sin embargo, como en la rama ELSE no hay sentencias, con 0 ejecuciones tenemos el 100%.

Para afrontar estos casos, se plantea un refinamiento de la cobertura de segmentos consistente en recorrer todas las posibles salidas de los puntos de decisión.

La ejecución de pruebas de caja blanca puede llevarse a cabo con un depurador (que permite la ejecución paso a paso), un listado del módulo y un rotulador para ir marcando por donde vamos pasando. Esta tarea es muy tediosa, pero puede ser automatizada. Hay compiladores que a la hora de generar código máquina dejan incrustado en el código suficiente código como para poder dejar un fichero (tras la ejecución) con el número de veces que se ha ejecutado cada sentencia, rama, bucle, etc.

Para nuestro sistema se ha logrado depurar con éxito líneas de código que se encuentren duplicadas o código que se halle obsoleto para el funcionamiento eficaz del sistema, pero que no implica la falla total del mismo.

4.5.3 PRUEBA DEL SISTEMA

Con esta prueba se crea directamente un análisis del sistema con los usuarios, el sistema debe tener un fácil uso, interacción con el usuario y sobre todo generar mensajes de control que sirvan de aviso y guía cuando el usuario no está realizando correctamente una tarea.

Con los usuarios del Centro de Salud No. 4 La Tola – Vicentina lograremos evaluar si su nivel de capacitación es óptima para la utilización eficaz del sistema y así poder crear políticas y lineamientos

del sistema. De la misma manera, sacar conclusiones de la forma en que los usuarios administradores del sistema tienen la capacidad de dar soporte y mantenimiento al Sistema de Gestión de Medicamentos SGMAS.

El control de este tipo de eventualidades y de mensajes que se realiza mediante scripts de verificación y validación de datos ingresados y acciones que necesiten mostrar errores al usuario cuando se está usando el sistema de una manera incorrecta.

Para las interfaces se han realizado ayudas visuales para la verificación de los mismos y cada uno de los campos que conforman el formulario va directamente direccionado a la verificación del script correspondiente para cada uno de los campos y formularios.

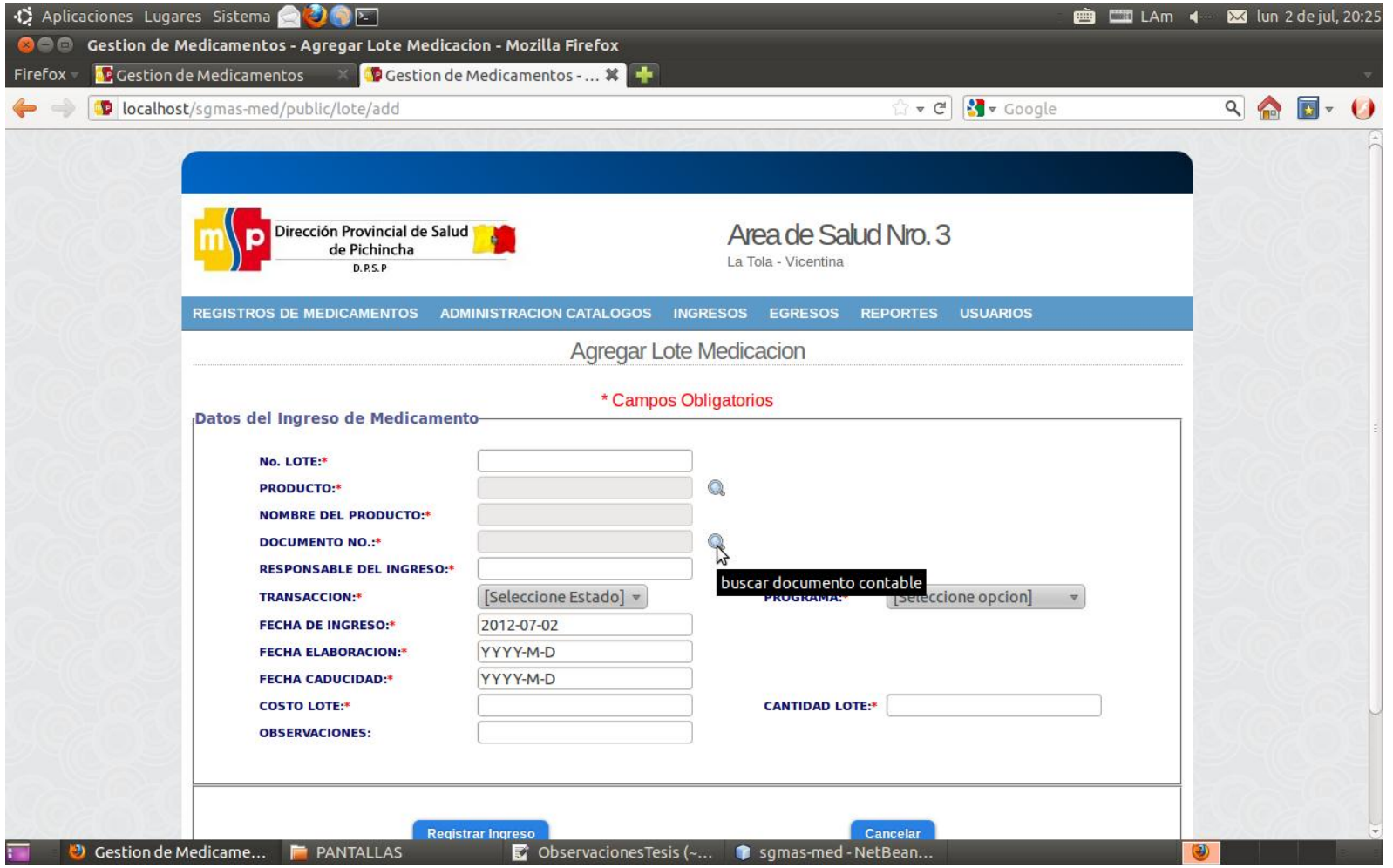


Figura 4.5.3.1 Prueba del Sistemas: Pantalla Agregar Lote Medicación SGMAS[89]

[89] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Figura 4.5.3.2 Prueba del Sistemas: Pantalla Registro Documento Contable SGMAS[90]

[90] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

CONCLUSIONES:

- Se construyó el módulo de Gestión de Medicamentos para el Sistema De Gestión Médico Para Áreas De Salud (Sgmas), utilizando una plataforma de libre distribución (Linux, PHP, PostgreSQL), en base a un conjunto de requerimientos proporcionados por el usuario, los mismos que fueron implementados en su totalidad.

- Las historias de Usuario son de mucha utilidad al momento de comprender la forma en que el sistema debe comportarse, ya que provee de la información de los requerimientos principales desde el punto de vista del usuario.

- El desarrollo de esta aplicación nos ha servido para reforzar los conocimientos proporcionados por la Universidad Politécnica Salesiana, en el transcurso de los diferentes niveles por los que hemos cursado.

- Esta herramienta informática permite a la Dirección Provincial de Salud, por intermedio de cada uno de los Subcentros de Salud, en los que se va a implementar, proporcionar un mejor servicio para los pacientes que requieran ser atendidos, ya que facilita la gestión adecuada de la entrega de medicinas.

RECOMENDACIONES:

- En el desarrollo del software es recomendable mantener una continua interacción con el usuario, como lo recomienda XP, ya que es el indicado para proporcionar al analista una retroalimentación conforme avanza en la construcción, de esta manera los objetivos del proyecto se mantendrán y se tendrá una idea clara de los aspectos que tienen que probarse durante el periodo de pruebas.

- Se recomienda realizar las pruebas del sistema, cada vez que se haya terminado algún módulo, para que no exista generación de código innecesario o algún inconveniente en su funcionalidad.

- Se necesita proporcionar a los estudiantes de la Universidad Politécnica Salesiana capacitación en herramientas de desarrollo de aplicaciones web, pues al momento solo poseemos conocimientos sobre el lenguaje de programación universal PHP y HTML.

- Se requiere que proporcionen de seminarios o se ponga en el pensum académico materias acerca del uso de programas de distribución libre, pues consta en el registro oficial la utilización de estos programas en cada institución, principalmente en las del Estado.

- Se necesita la creación de convenios interinstitucionales con las empresas que requieran de nuestros servicios, para que podamos aplicar nuestros conocimientos y reforzar con la práctica, y se nos proporcione alguna remuneración económica que resultaría de gran ayuda para nosotros como estudiantes.

BIBLIOGRAFÍA:

- LOPEZ QUIJADO José, *Domine PHP 5 Editorial Ra-ma,* Madrid 2008
- JACOBSON Ivar, *El Proceso Unificado de desarrollo de Software*, Editorial Pearson Education, Madrid 2000
- PEREZ LOPEZ Cesar, *Administración de sitios y páginas web con Macromedia Dreamweaver 8*, Editorial Alfaomega, México 2007
- PFLEEGER Shari, *Ingeniería de software teoría y práctica*, Editorial Pearson Education, Buenos Aires 2002
- SOMMERVILLE Ian, *Ingeniería de software*, Editorial Pearson Education, México 2002
- BACK A., *Introducción a la programación de sistemas*, Editorial Addison Wesley, México 2000
- GIL RUBIO Javier, *Creación de páginas web con PHP 4*, Editorial McGraw Hill, Madrid 2001
- PEREZ LOPEZ Cesar, *MySQL para Windows y Linux*, Editorial Alfaomega, México 2004
- HOINZER Steven, *PHP Manual de referencia*, Editorial McGraw Hill, México 2009
- BUYENS Jim, *Aprenda desarrollo de base de datos*, Editorial McGraw Hill, Madrid 2001
- LOPEZ QUIJADO José, *Domine PHP, .NET y MySQL*. Programación dinámica en el lado del servidor, Editorial Ra-ma, Madrid 2007
- KILIAN Crawford, *Escribir para la web,* Editorial Debusto, Bilbao 2001
- LARMAN Craig, *UML y patrones*, Editorial Pearson Education, Madrid 2003
- BRUEGGE Bernd, *Ingeniería de software orientado a software*, Editorial Pearson Education, México 2002
- BOOCH Grady, *El lenguaje unificado de modelado UML*, Editorial Addison Wesley, México 1999
- FOWLER Martin, *UML gota a gota*, Editorial Addison Wesley, México 1999

- STEVENS Perdita, *Utilización de un UML en Ingeniería del Software con objetos y componentes*, Editorial Pearson Education, Madrid 2002
- DORSEY Paul, *Diseño de bases de datos con UML*, Editorial McGraw Hill, Madrid 1999
- KENDALL Kenneth, *Análisis y Diseño de Sistemas*, Editorial Pearson Education, México 1997.
- RUMBAUGH James, El lenguaje unificado de modelado, Editorial Pearson Education, Madrid 2000 (PDF)

ANEXOS

UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO-CAMPUS SUR

CARRERA DE INGENIERÍA DE SISTEMAS

MENCIÓN TELEMÁTICA

ANÁLISIS, DISEÑO Y DESARROLLO DE UN MÓDULO PARA LA GESTIÓN DE MEDICAMENTOS DEL SISTEMA DE GESTIÓN MÉDICO PARA ÁREAS DE SALUD (SGMAS), PARA EL CENTRO DE SALUD NO. 3 "LA TOLA-VICENTINA" DE LA DIRECCIÓN PROVINCIAL DE SALUD DE PICHINCHA.

GUÍA DEL PROGRAMADOR

CARLOS FERNANDO ARIAS MONTALVO
MARÍA JOSÉ MICHELENA PERUGACHI

DIRECTOR ING. RENÉ ARÉVALO

Quito, Octubre del 2012

GUIA DEL PROGRAMADOR

TABLA DE CONTENIDOS

1. INTRODUCCIÓN A ZEND FRAMEWORK

1.1 DESCRIPCIÓN GENERAL

Zend Framework (ZF) es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP5 así como es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de Zend Framework es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. A menudo se refiere a este tipo de diseño como "use-at-will" (uso a voluntad).

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend Framework conforman un potente y extensible framework de aplicaciones web al combinarse. ZF ofrece un gran rendimiento y una robusta implementación MVC (MODELO VISTA CONTROLADOR), una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. Cualesquiera que sean las necesidades de su solicitud, usted tiene todas las posibilidades de encontrar un componente de Zend Framework que se pueda utilizar para reducir drásticamente el tiempo de desarrollo.

1.1.1 INSTALACIÓN

Descarga[1] la última versión de Framework desde el sitio oficial.

[1] http://framework.zend.com/download/latest7

Cuando esta descargada la aplicación creamos una estructura de directorios, podemos crearla automáticamente con Zend_Tool_Framework, o puedes hacerlo manualmente.

La estructura inicial debe quedar de la siguiente forma:

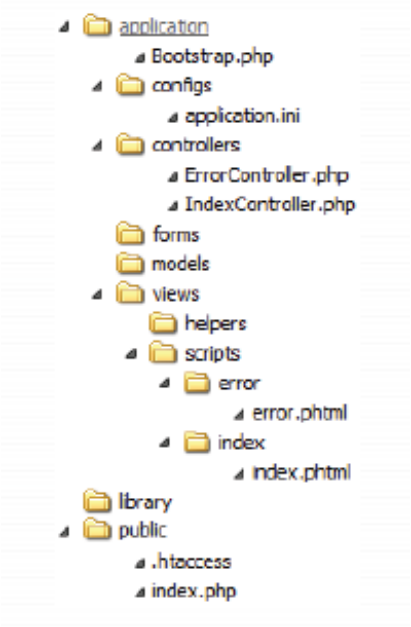


Figura 1.1.1. Pantalla Principal. [91]

1.1.2 ACTION CONTROLLERS

Los controladores son clases que extienden de Zend_Controller_Action cada controlador tiene unos métodos especiales cuya nombre tiene el sufijo “Action” y denominados “action methods”.

[91] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Por default las URLs en Zend Framework son del tipo /controlador/action es decir que si en nuestro ProductoController tenemos un “IndexAction” lo podremos ejecutar desde /producto/index.

La estructura es simple en el método init() se agregan tareas de inicialización y creamos un action llamado index.

Al ejecutarlo, Zend Framework automáticamente relaciona el nombre del action con una vista que será renderizada.

Esta vista tiene la extensión phml en el caso del indexAction la vista asociada será index.phtml dentro de la carpeta correspondiente al controlador index en views/scripts.

Para nuestro ejemplo sería /views/scripts/producto/index.phtml

1.1.3 CREACIÓN DE PLANTILLA PRINCIPAL

El componente de Zend Framework que da soporte a las vistas es Zend_View. Este nos permite separar la presentación de la lógica o reglas de negocio que se encuentra en el modelo.

En Zend Framework encontramos las vistas en el directorio “views” y se organizan de la forma:

views/scripts/{nombredelcontrolador}/{nombredelaaccion}.phtml.

En la mayoría de los proyectos hay partes de código HTML que se repite para todas las vistas, por ejemplo: un encabezado, una columna lateral y el pie de página. Para evitar repetir código es que existe la posibilidad de crear un “layout” o plantilla donde colocaremos el código común y desde donde llamaremos las vistas.

Lo primero que debemos hacer es crear el directorio: “application/layouts/” y agregar en nuestro archivo de configuración la siguiente el siguiente código HTML que corresponde a la plantilla creada para la realización del proyecto y acorde a las exigencias del Centro de Salud La Tola – Vicentina y el Ministerio Salud Pública para ello modificamos el archivo application.ini el la siguiente línea:

resources.layout.layoutpath = APPLICATION_PATH “/layouts”

En la cual colocamos dentro de la sección [production] del archivo de configuración de la aplicación.

Además, necesitamos inicializar nuestra vista en la clase Boostrap, para lo que crearemos otros métodos que comiencen por el prefijo _init justo debajo de _initAutoload(). El método se llamará _initViewHelpers()

```
<?php

class Bootstrap extends Zend_Application_Bootstrap_Bootstrap
{

  protected function _initAutoload(){
   $moduleLoader = new
Zend_Application_Module_Autoloader(array('namespace' => '', 'basePath' =>
PLICATION_PATH));
   return $moduleLoader;
  }

     protected function _initView()
  {

    $this->bootstrap('layout');
```

```
        $layout = $this->getResource('layout');
        $view = $layout->getView();
        $view = new Zend_View();
        $view->setEncoding('UTF-8');
        $view->doctype('XHTML1_STRICT');
        $view->headMeta()->appendHttpEquiv('Content-Type',
'text/html;charset=utf-8');

         //Agrego el path de ZendX para poder usar JQuerry
        $view->headTitle()->setSeparator(' - ');
        $view->headTitle('Gestion de Medicamentos');
        $view->addHelperPath('ZendX/JQuery/View/Helper/',
'ZendX_JQuery_View_Helper');
        $viewRenderer                                                    =
Zend_Controller_Action_HelperBroker::getStaticHelper('ViewRenderer');
        $viewRenderer->setView($view);
        return $view;
    }

    protected function _initViewHelpers()
    {
      $this->bootstrap('layout');
      $layout = $this->getResource('layout');
      $view = $layout->getView();
      $view = new Zend_View();
      $view->setEncoding('UTF-8');
      $view->doctype('XHTML1_STRICT');
      $view->headMeta()->appendHttpEquiv(
                        'Content-Type', 'text/html;charset=utf-8'
      );

      $view->headTitle()->setSeparator(' - ');
      $view->headTitle('Gestion de Medicamentos');
```

```
    $viewRenderer = Zend_Controller_Action_HelperBroker::getStaticHelper(
            'ViewRenderer'
    );
    $viewRenderer->setView($view);

    Zend_Session::start();

    return $view;

  }
}
```

A través del método bootstrap() inicializamos la vista para cuando después recuperemos el objeto Zend_Layout a través de getResource(), para después recuperar la vista utilizando el método getView().

Una vez que tenemos la vista instanciada, podemos llamar a varios métodos de ayuda de la vista, que definirán en nuestra vista el código correspondiente para cada uno de ellos.

El código que tendremos en nuestra vista “layout.phtml” y que tendremos en el directorio /application/layouts sería el siguiente para todos nuestros formularios que deseemos aplicar la plantilla creada:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<!--<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />-->

<?php echo $this->headMeta(); ?>
<?php echo $this->headTitle(); ?>
```

```
<?php echo $this->headScript(); ?>
<?php echo $this->headLink(); ?>
<!-- Menu plantilla-->
<link href="<?php echo $this->baseUrl; ?>/sgmas-med/public/css/style.css"
media="screen" rel="stylesheet" type="text/css" />
<link href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/css/dropdown/dropdown.css" media="screen" rel="stylesheet"
type="text/css" />
<link href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/css/dropdown/themes/nvidia.com/default.advanced.css"
media="screen" rel="stylesheet" type="text/css" />
<link href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/images/favicon(1).ico" type="image/x-icon" rel="shortcut icon" />
<link href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/js/jquery/jquery-
1.4.3.js" type="image/x-icon" rel="shortcut icon" />
<!--<link rel="icon" type="image/png"
href="images/themes/default/Introduction.png" />-->
<!--[if lt IE 7]>
<script type="text/javascript" src="js/jquery/jquery.js"></script>
<script type="text/javascript" src="js/jquery/jquery.dropdown.js"></script>
<![endif]-->


</head>

<body>
<div id="wrap">
  <div id="cabecera">

   <?php if($this->user) : ?>
       <p style="color: white; font-size: 10.5px" id="logged-in"
```

```
align='right'><?php
          echo $this->escape($this->user->perfil);?> |  
         <img src="<?php echo $this->baseUrl; ?>/images/logout.png"
alt="cerrar sesion" ></img>
        <a style="color: white;" href="<?php echo $this->baseUrl
?>/index/logout">cerrar sesiÃ³n</a>
              
        </p>
    <?php endif; ?>
  </div>
   <div id="header">
 <table>
   <tr>
          <td><div id="logo"> </div></td>
          <td align = "right" ><h1>Area de Salud Nro. 3</h1> <h2>La Tola –
Vicentina</h2> </td>
   </tr>
 </table>
</div>

<div id="menu">
<ul class="dropdown dropdown-horizontal" >
  <li><a href="./" class="dir">REGISTROS DE MEDICAMENTOS</a>
          <ul>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/producto">Productos</a></li>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/ubicacion">Ubicaci&oacute;n F&iacute;sica</a></li>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/bajas">Registro de Bajas</a></li>

<!--              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/lote">Lotes de medicaci&oacute;n</a></li>-->
```

```
            </ul>
  </li>
  <li><a href="./" class="dir">MANTENIMIENTO</a>
          <ul>
              <li><a href="./">PRODUCTOS</a></li>
              <li><a href="./"></a>.....</li>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/bodega">Bodega</a></li>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/familia">Familia</a></li>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/presentacion">Presentaci&oacute;n</a></li>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/responsable">Responsables</a></li>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/unidades">Unidades</a></li>
              <li><a href="./"></a>.....</li>
              <li><a href="./">LOTE</a></li>
              <li><a href="./"></a>.....</li>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/proveedores">Proveedores</a></li>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/tipo">Documento Contable</a></li>
              <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/documento">Registro Documento Contable</a></li>

            </ul>
  </li>
      <li><a href="<?php echo $this->baseUrl; ?>/sgmas-med/public/lote"
class="dir">INGRESOS</a>
      </li>
  <li><a href="<?php echo $this->baseUrl; ?>/sgmas-med/public/despachos"
class="dir">EGRESOS</a>
```

```
<!--            <ul>
                <li><a href="<?php //echo $this->baseUrl; ?>/sgmas-
med/public/despachos">Despachos</a></li>
                <li><a href="./">Opcion 4C</a></li>
                <li><a href="./">Opcion 4D</a></li>
                <li><a href="./">Opcion 4E</a></li>
                <li><a href="./">Opcion 4F</a></li>
                <li><a href="./">Opcion 4G</a></li>
                <li><a href="./">Opcion 4H</a></li>
        </ul>-->
  </li>
  <li><a href="./" class="dir">REPORTES</a>
        <ul>
                <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/cantidad">Stock lotes</a></li>
                <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/preciounitario">Precio Unitario Promedio</a></li>
                <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/despachos/imprimirdespachos">Imprimir Despachos</a></li>
                <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/reportedespachosperiodos/">Reporte Periodos
Despachos</a></li>
                <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/registro-contable/">Reporte Registro Contable</a></li>
                <li><a href="<?php echo $this->baseUrl; ?>/sgmas-
med/public/control-ingresos/">Control Ingresos</a></li>
<!--                    <li><a href="./">  </a></li>
                <li><a href="./">  </a></li>-->
        </ul>
  </li>
  <li><a href="./" class="dir">USUARIOS</a>
<!--            <ul>
                <li><a href="./">Opcion 6A</a></li>
```

```
                <li><a href="./">Opcion 6B</a></li>
                <li><a href="./">Opcion 6C</a></li>
                <li><a href="./">Opcion 6D</a></li>
                <li><a href="./">Opcion 6E</a></li>
                <li><a href="./">Opcion 6F</a></li>
                <li><a href="./">Opcion 6G</a></li>
                <li><a href="./">Opcion 6H</a></li>
                <li><a href="./">Opcion 6I</a></li>
            </ul>-->
  </li>
</ul>

<!-- / END -->

<!--        <li><a href="#">Page1</a></li> -->
<!--        <li><a href="#">Page2</a></li> -->
<!--        <li><a href="#">Page3</a></li> -->
<!--        <li><a href="#">Page4</a></li> -->

</div>

<div id="content">

<center>
<?php echo $this->escape($this->title); ?>
<?php echo $this->layout()->content; ?>
</center>

<div style="clear:both;">

</div>
</div>
</div>
  <div id="footer">
```

```
            <h2>UNIVERSIDAD POLITECNICA SALESIANA</h2>
            <h2>Ingenieria en Sistemas - Telematica / Robotica </h2>
               <h2><a href="<?php echo $this->baseUrl ?>/sgmas-
med/public/auth/logout">Salir del Sistema</a></h2>
  </div>
</div>

</body>
</html>
```

En el código de nuestra página, vemos como tenemos una instancia del objeto de la vista a través de $this, por lo que podemos utilizar esta instancia para llamar a los métodos asociados a la vista y recuperar información que le ha sido asignada a la vista.

1.1.4 CONSTRUCCIÓN DE FORMULARIOS

La construcción de los formularios en zend framework del sistema se los realiza en script.phtml con esto podemos tener el control de toda la maquetación del formulario así como de las validaciones de todos los campos que se coloquen en el formulario dependiendo de las necesidades del sistema siendo estas más sencillas y que requieren de un menor grado de seguridad, al ser algoritmos conocidos por todo el mundo, por ejemplo, validación de campo vacío, validación de e-mail, validación de que todos los caracteres del campo son numéricos, etc.

Se crea la clase Form donde seguiremos construyendo el formulario y añadiendo las validaciones necesarias que deban ser realizadas a nivel de

servidor y así tendremos una vista donde pondremos el resultado final del proceso realizado.

1.1.5 CONSTRUCCIÓN DE INTERFACES

Para la construcción de cada una de las interfaces del sistema se ha creado cada formulario en lenguaje PHP con su respectiva extensión que se guardan en la carpeta correspondiente a los forms en el proyecto de zend framework con el nombre de cada interfaz que compone el sistema debidamente identificados.

Uno de los métodos utilizados es el Zend_Form que es el componente de Zend Framework encargado de crear y manejar los formularios que usemos en nuestras aplicaciones para la construcción del sistema, por lo que es un componente de uso casi obligado en cualquier desarrollo web y uno de los que más llaman la atención al empezar a trabajar con este framework. Las validaciones que proporciona esta herramienta se los hace a nivel del servidor por lo que resulta una carga al mismo lo que nos lleva a la solución de aplicar como se dijo anteriormente la utilización de javascripts para realizar esa función pues a diferencia de lo antes expuesto estas validaciones se las hace a nivel del cliente.

El siguiente código es la construcción de la interfaz para el manejo de los medicamentos con las entradas de datos correspondientes a la información general cada uno de los productos ingresados al sistema.

Para el ejemplo se utiliza el método Zend_Form con sus correspondientes validaciones.

```
<?php
class Application_Form_Producto extends Zend_Form
{
    public function init()
```

```
{
    $this->setName('producto');

    // campo hidden para guardar id del producto
    $id_producto = new Zend_Form_Element_Hidden('id_producto');
    $id_producto->addFilter('Int');

    $id_bodega = new Zend_Form_Element_Select('id_bodega');
    $id_bodega->setLabel('')->setRequired(true);

    // carga en un select los tipos de bodega existentes en Base de Datos
    $table = new Application_Model_DbTable_Bodega();

    //obtengo listado de todas las bodegas y los recorro en un
    //arreglo para agregarlos a la lista
    foreach ($table->listar() as $c)
    {
       $id_bodega->addMultiOption($c->id_bodega, $c->nombre_bodega);
    }

    $id_familia = new Zend_Form_Element_Select('id_familia');
    $id_familia->setLabel('')->setRequired(true);

    $table2 = new Application_Model_DbTable_Familia();
    foreach ($table2->listar() as $c1)
    {
       $id_familia->addMultiOption($c1->id_familia, $c1->nombre_familia);
    }

    $id_presentacion = new
Zend_Form_Element_Select('id_presentacion');
    $id_presentacion->setLabel('')->setRequired(true);
    $table3 = new Application_Model_DbTable_Presentacion();
```

```
foreach ($table3->listar() as $c2)
{
     $id_presentacion->addMultiOption($c2->id_presentacion,        $c2-
     >nombre_presentacion);
}

$id_unidad = new Zend_Form_Element_Select('id_unidad');
$id_unidad->setLabel('')->setRequired(true);
$table4 = new Application_Model_DbTable_Unidades();
foreach ($table4->listar() as $c3)
{
   $id_unidad->addMultiOption($c3->id_unidad, $c3->unidad_medida);
}

$id_comercial = new Zend_Form_Element_Select('id_comercial');
$id_comercial->setLabel('')->setRequired(true);

$table5 = new Application_Model_DbTable_Comercial();
foreach ($table5->listar() as $c4)
{
 $id_comercial->addMultiOption($c4->id_comercial,$c4-
>presentacion_comercial);
}

//creamos <input text> para escribir nombre album
$nombre_producto = new
  Zend_Form_Element_Text('nombre_producto');

$nombre_producto->setLabel('Nombre del producto:')
//VALIDACIONES DE ZEND
       ->setRequired(true) // CAMPO OBLIGATORIO
       ->addFilter('StripTags')
       ->addFilter('StringTrim') // ELIMINACION DE CAMPOS BLANCO
```

```
            ->addFilter('StringToUpper') // CONVERSION MAYUSCULAS
            ->addValidator('NotEmpty'); // CAMPO NO DEBE ESTAR EN
                                        // BLANCO

        $estado_producto = new
Zend_Form_Element_Text('estado_producto');
        $estado_producto->setLabel('')
            ->setRequired(false)
            ->addFilter('StripTags')
            ->addFilter('StringToUpper')
            ->addFilter('StringTrim');

        $observacion_producto = new
          Zend_Form_Element_Text('observacion_producto');
        $observacion_producto->setLabel('')
            ->setRequired(false)
            ->addFilter('StripTags')
            ->addFilter('StringToUpper')
            ->addFilter('StringTrim');

        $stock_max = new Zend_Form_Element_Text('stock_max');
        $stock_max->setLabel('')
            ->setRequired(false)
            ->addFilter('StripTags')
            ->addValidator('Digits')
            ->addValidator('NotEmpty')
            ->addFilter('StringTrim');

        $stock_min = new Zend_Form_Element_Text('stock_min');
        $stock_min->setLabel('')
            ->setRequired(false)
            ->addFilter('StripTags')
            ->addValidator('Digits')
```

```
            ->addValidator('NotEmpty')
            ->addFilter('StringTrim');


        //boton para enviar formulario
        $submit = new Zend_Form_Element_Submit('submit');
        $submit->setAttrib('id', 'submitbutton');

        $this->addElements(array($id_producto, $id_bodega,
        $id_familia, $id_presentacion, $id_unidad,
        $id_comercial,$nombre_producto, $estado_producto,
        $observacion_producto, $stock_max, $stock_min, $submit));
    }
}
```

Con este método se le carga al servidor todas validaciones del formulario que se pueden hacer a nivel de cliente así como la imposibilidad de crear formularios cuyos campos no sean secuenciales.

De la misma forma no se puede personalizar los formularios con la utilización de "decorators" que nos permiten "customizar" los formularios y darle el estilo que queramos, en muchas ocasiones significa introducir código HTML en el código de servidor; es por tal motivo que nuestro objetivo siempre es separar claramente las capas de negocio y la de presentación, y aunque esta segunda tenga la clase "Form" en ella no es aconsejable introducir código que pertenezca a la vista. Por tal motivo realizamos cambios en la vista del proyecto e introducimos código HTML de manera que podamos modificar y manipular el estilo del formulario a la medida que los formularios sean de fácil manipulación para el usuario y para el eficiente trabajo del servidor brindándole más agilidad para la resolución de las actividades para lo que fue encomendado.

Para ello modificamos el archivo que zend crea simultáneamente en el directorio: /views/producto, con el mismo nombre del Form creado introduciendo lenguaje HTML y creando archivos javascripts para la validación de los campos del mismo formulario.

En el siguiente código vemos como podemos realizar formularios modificando la vista del directorio: /views/producto/add.phtml, utilizando HTML como lenguaje modelador de la interfaz para hacerlo más atractivo a la vista del usuario así:

```
<h3><center><?php echo $this->escape($this->title);
?></center></h3><BR>

<center>
   <div id="fondogrid">
<form action="" id="producto" onsubmit="return validatefields();"
method="post" name="producto">

   <p style="color: red; font-family: bold, Arial; font-size: 16px">* Campos
Obligatorios</p>

   <fieldset><legend>Datos del Medicamento</legend><br>

      <table style="text-align: left; width: 765px; height: 80px; vertical-align:
middle;"
            border="0">
         <tbody>
            <tr>
               <td>
                  <label>CUM:</label><label style="color: red">*</label>
               </td>
               <td>
```

```
                    <input name="observacion_producto"
id="observacion_producto"  class="mayusculas" size="20" type="text"/>
                </td>
              </tr>
              <tr>
                <td>
                    <label>NOMBRE DE LA BODEGA:</label><label
style="color: red">*</label>
                </td>
                <td>
            <select name="id_bodega" id="id_bodega" class="listc">
             <option value="0">[Seleccione opcion]</option>
                <?php foreach ($this->bodega as $c) : ?>
                    <?php if ($c->id_bodega == $this->id_bodega) { ?>
                      <option value="<?php echo $this->escape($c->id_bodega);
?>" selected><?php echo $this->escape($c->nombre_bodega); ?></option>
                    <?php }else { ?>
                      <option value="<?php echo $this->escape($c->id_bodega);
?>"><?php echo $this->escape($c->nombre_bodega); ?></option>
                    <?php } ?>
                <?php endforeach; ?>
            </select>
                </td>
                <td>
                    <label>FAMILIA DEL PRODUCTO:</label><label
style="color: red">*</label>
                </td>
                <td>
          <select name="id_familia" id="id_familia" class="listc">
            <option value="0">[Seleccione opcion]</option>
                <?php foreach ($this->familia as $c) : ?>
                    <?php if ($c->id_familia == $this->id_familia) { ?>
```

```
                    <option value="<?php echo $this->escape($c->id_familia);
?>" selected><?php echo $this->escape($c->nombre_familia); ?></option>
                <?php }else { ?>
                    <option value="<?php echo $this->escape($c->id_familia);
?>"><?php echo $this->escape($c->nombre_familia); ?></option>
                <?php } ?>
            <?php endforeach; ?>
        </select>
            </td>
        </tr>
        <tr>
            <td>
                <label>FORMA FARMACEUTICA:</label><label style="color:
red">*</label>
            </td>
            <td>
        <select name="id_presentacion" id="id_presentacion" class="listc">
          <option value="0">[Seleccione opcion]</option>
            <?php foreach ($this->presentacion as $c) : ?>
                <?php if ($c->id_presentacion == $this->id_presentacion) {
?>
                    <option value="<?php echo $this->escape($c-
>id_presentacion); ?>" selected><?php echo $this->escape($c-
>nombre_presentacion); ?></option>
                <?php }else { ?>
                    <option value="<?php echo $this->escape($c-
>id_presentacion); ?>"><?php echo $this->escape($c-
>nombre_presentacion); ?></option>
                <?php } ?>
            <?php endforeach; ?>
        </select
            </td>
            <td>
```

```
                    <label>CONCENTRACION:</label><label style="color:
red">*</label>
                </td>
                <td>
          <select name="id_unidad" id="id_unidad" class="listc">
            <option value="0">[Seleccione opcion]</option>
                <?php foreach ($this->unidad as $c) : ?>
                    <?php if ($c->id_unidad == $this->id_unidad) { ?>
                      <option value="<?php echo $this->escape($c->id_unidad);
?>" selected><?php echo $this->escape($c->unidad_medida); ?></option>
                    <?php }else { ?>
                      <option value="<?php echo $this->escape($c->id_unidad);
?>"><?php echo $this->escape($c->unidad_medida); ?></option>
                    <?php } ?>
                <?php endforeach; ?>
           </select
                </td>

              </tr>
               <tr>
                <td>
                    <label>NOMBRE DEL PRODUCTO:</label><label
style="color: red">*</label>
                </td>
                <td>
                    <input id="nombre_producto" class="mayusculas"
maxlength="20" size="20" tabindex="3" name="nombre_producto">
              </tr>
              <tr>
                <td>
                    <label>ESTADO DEL PRODUCTO:</label><label
style="color: red">*</label>
                </td>
```

```
                <td>
                    <select name="estado_producto" id="estado_producto">
                        <option value="NINGUNO">[Seleccione Estado]</option>
                        <option value="RESAGADO">RESAGADO</option>
                        <option
value="DESCONTINUADO">DESCONTINUADO</option>
                        <option
value="DETERIORADO">DETERIORADO</option>
                    </select>
<!--                    <input id="estado_producto" maxlength="20" size="20"
tabindex="3" name="estado_producto">-->
                </td>
            </tr>
        </tbody>
    </table>
    <table style="text-align: left; width: 765px; vertical-align: middle;"
        border="0">
        <tbody>
            <tr>
                <td>
                    <label>PRESENTACION COMERCIAL:</label><label
style="color: red">*</label>
                </td>
                <td>
        <select name="id_comercial" id="id_comercial" class="listc"
style="text-align: left;">
            <option value="0">[Seleccione opcion]</option>
                <?php foreach ($this->comercial as $c) : ?>
                    <?php if ($c->id_comercial == $this->id_comercial) { ?>
                      <option value="<?php echo $this->escape($c-
>id_comercial); ?>" selected><?php echo $this->escape($c-
>presentacion_comercial); ?></option>
                    <?php }else { ?>
```

```
                    <option value="<?php echo $this->escape($c-
>id_comercial); ?>"><?php echo $this->escape($c-
>presentacion_comercial); ?></option>
                    <?php } ?>
                <?php endforeach; ?>
            </select
                </td>
                </tr>
              </tbody>
        </table>
        <table style="text-align: left; width: 765px; vertical-align: middle;"
             border="0">
            <tbody>
              <tr>
                <td>
                  <label>STOCK MAXIMO:</label><label style="color:
red">*</label>
                </td>
                <td>
                  <input name="stock_max" id="stock_max"
class="mayusculas" size="20" type="text" onkeypress = "return
validarNume(event)"/>
                </td>
                <td>
                  <label>STOCK MINIMO:</label><label style="color:
red">*</label>
                </td>
                <td>
                  <input name="stock_min" id="stock_min"
class="mayusculas" size="20" type="text" onkeypress = "return
validarNume(event)"/>
                </td>
              </tr>
```

```
            </tbody>
        </table>
        <br><br>
    </fieldset>
    <fieldset>
        <br>
        <table style="text-align: center; width: 765px; height: 52px;"
            border="0" cellpadding="0" cellspacing="0">
            <tbody>
                <tr>
                    <td style="vertical-align: middle; width: 50%;">
                        <input onclick="sendform()" class="button small blue"
                            value="Registrar Producto" type="button"></td>
                            <td style="vertical-align: middle; width: 50%;">
                                <input type="button" class="button small blue"
onClick="document.location = '<?php echo $this->url(array('controller' =>
'producto',
            'action' => 'index'));?>'" name="Cancelar" value="Cancelar">
                                <br>
                            </td>
                </tr>
            </tbody>
        </table>
        <br>
    </fieldset>
</form>
    </div>
</center>
<BR>
<div id = "errores"></div>
```

De la misma manera suprimiremos todo lo relacionado con el botón "submit", para tener el control de cuándo se envían los datos del formulario al servidor

con esto eliminaremos las validaciones que hacemos a nivel de cliente de nuestra clase Form, ya que esa función la controlamos desde nuestro javascript, una vez se hayan pasado todas las validaciones a nivel de cliente.

El método constructor de nuestra clase Producto, quedaría de la siguiente manera en el directorio: /public/js/validación_producto.js:

```
merror = "";

String.prototype.trim = function() {
        return this.replace(/^\s+|\s+$/g, "");
};

function validarNume(e)
{
   tecla = (document.all) ? e.keyCode : e.which;
   if (tecla == 8 || e.keyCode == 9 ) return true;
   patron = /\d/;
   te = String.fromCharCode(tecla);
   return patron.test(te);
}

function noNumero(valor){
   //Comprobamos si es número
   if (/^([0-9])*$/.test(valor))
      return false;
   else
      return true;
}

function validatefields(){
   componentes = document.getElementById("producto").elements;
   error = false;
```

```
for (i=0; i< componentes.length; i++){
  switch (componentes[i].name){

    case "stock_max" :if (componentes[i].value.trim().length == 0){
                merror = "El campo stock maximo es obligatorio";
                error = true;
                    document.producto.stock_max.focus()
              }
               break;

    case "stock_min" :if (componentes[i].value.trim().length == 0){
                merror = "El campo stock minimo es obligatorio";
                error = true;
                    document.producto.stock_min.focus()
              }
               break;

    case "observacion_producto" :if (componentes[i].value.trim().length ==
          0){
                merror = "El campo CUM es obligatorio";
                error = true;
                    document.producto.observacion_producto.focus()
              }
               break;


    case "nombre_producto" :if (componentes[i].value.trim().length == 0){
                merror = "El campo nombre producto es obligatorio";
                error = true;
                    document.producto.nombre_producto.focus()
              }
               break;
```

```
        }
    }

     return (!error);
}

function eliminar_producto(id, producto){

   var confir = confirm("¿Desea eliminar el registro del producto " + producto+" ?") ;
   if(confir)
   {
       $.ajax(
     {
          dataType: "html",
          type: "POST",
          url: "/sgmas-med/public/producto/delete", // ruta donde se encuentra nuestro action que procesa la peticion XmlHttpRequest
          data: 'id_producto='+ id , //Se añade el parametro de busqueda por nombre
          beforeSend: function(data){
             },
          success: function(requestData){   //Llamada exitosa
             alert("Registro eliminado correctamente");
             document.location = '/sgmas-med/public/producto';
           },
           error: function(requestData, strError, strTipoError){
           alert("Error " + strTipoError +': ' + strError ); //En caso de error mostraremos un alert
          },
          complete: function(requestData){          //Llamada exitosa
          document.location = '/sgmas-med/public/producto';
          }
```

```
        });

    }
}

function actualizar_producto(){

   if (validatefields()){
      var id_producto = document.producto.id_producto.value;
      var id_bodega = document.producto.id_bodega.options[document.producto.id_bodega.selectedIndex].value;
      var id_familia = document.producto.id_familia.options[document.producto.id_familia.selectedIndex].value;
      var id_presentacion = document.producto.id_presentacion.options[document.producto.id_presentacion.selectedIndex].value;
      var id_unidad = document.producto.id_unidad.options[document.producto.id_unidad.selectedIndex].value;
      var id_comercial = document.producto.id_comercial.options[document.producto.id_comercial.selectedIndex].value;
      var nombre_producto = document.producto.nombre_producto.value;
      var estado_producto = document.producto.estado_producto.value;
      var observacion_producto = document.producto.observacion_producto.value;
      var stock_max = document.producto.stock_max.value;
      var stock_min = document.producto.stock_min.value;

//alert(id_producto+"-"+id_bodega+"-"+"-"+id_familia+"-"+"-"+id_presentacion+"-"+"-"+id_unidad+"-"+"-"+nombre_producto);
```

```
    $.ajax(
     {
         dataType: "html",
         type: "POST",
         url: "/sgmas-med/public/producto/update", // ruta donde se encuentra nuestro action que procesa la peticion XmlHttpRequest
         data: '&id_producto=' +id_producto + '&id_bodega='+ id_bodega + '&id_familia=' + id_familia + '&id_presentacion=' + id_presentacion + '&id_unidad=' + id_unidad + '&id_comercial=' + id_comercial + '&nombre_producto=' + nombre_producto + '&estado_producto=' + estado_producto + '&observacion_producto=' + observacion_producto + '&stock_max=' + stock_max + '&stock_min=' + stock_min, //Se añade el parametro de busqueda ya sea por nombre o por especialidad
         beforeSend: function(data){
            },
         success: function(requestData){   //Llamada exitosa
          alert("Datos actualizados correctamente");
          },
          error: function(requestData, strError, strTipoError){
          alert("Error " + strTipoError +': ' + strError ); //En caso de error mostraremos un alert
         },
         complete: function(requestData){ //Llamada exitosa
          document.location = '/sgmas-med/public/producto';
         }
      });

  }else
  {
    alert(merror);
   }
```

```
}

function sendform(){

    if (validatefields()){
        alert("Registro guardado correctamente");
        document.getElementById("producto").submit();}
    else{
        alert(merror);

        }
}
```

Cabe destacar que esta clase, aunque parezca que haya perdido su utilidad, no será así, ya que nos servirá para aplicar filtros o para hacer validaciones a nivel de servidor, pues así lograremos no hacer público el algoritmo de validación de cualquier dato.

La utilización de CSS en código HTML modificando las vistas de los formularios del sistema nos permite realizar formularios personalizados para dar estilos a las interfaces que se conviertan en una herramienta agradable a la vista del usuario y de fácil manipulación.

El archivo CSS se ubica en el directorio: /public/css/style.css que se invoca a la cabecera de cada archivo .phtm en la vista del formulario quedando de esta manera:

```
*
{
 padding: 0;
 margin: 0;
}
```

```
body
{
 background-color: #E7E8E9;
 background-image: url('../images/fondopage.jpg');
 background-repeat: repeat;
 background-position: left top;
 font-family: Arial;
 font-size: 12px;
 padding: 0;
 margin: 0;
 color: #555;
 line-height: 17px;
}

img
{
 border-style: none;
}

a
{
 color: #5C6F4C;
}

a:hover
{
 text-decoration: none;
 color: #999;
}

h3
{
 border-bottom-color: #aaa;
```

```
 border-bottom-width: 1px;
 border-bottom-style: dotted;
 color: #666;
 padding-bottom: 4px;
 margin-top: 0px;
 margin-right: 0;
 margin-bottom: 7px;
 margin-left: 0;
 font-weight: 100;
 font-size: 22px;
 letter-spacing: -1px;
}

h3 a
{
 text-decoration: none;
 font-size: 22px;
 letter-spacing: -1px;
}

h3 a:hover
{
 color: #999;
}

#wrap
{
 width: 940px;
 margin-top: 20px;
 margin-right: auto;
 margin-bottom: 20px;
 margin-left: auto;
}
```

```
#header
{
 background-color: #FFFFFF;
 font-family: "Century Gothic", "Trebuchet MS", "Arial Narrow", Arial, sans-serif;
 height: 95px;
}

#header h1
{
 font-size: 35px;
 font-size: 28px;
 font-weight: 100;
 letter-spacing: -3px;
 padding-top: 0;
 padding-right: 0;
 padding-bottom: 5px;
 padding-left: 23px;

 text-align: left;
}

#header h1 a
{
 color: #5C6F4C;
 text-decoration: none;
}

#header h1 a:hover
{
 color: #111;
 text-decoration: none;
```

```
}

#header h2
{
 font-size: 13px;
 color: #666;
 padding-top: 4px;
 padding-right: 0;
 padding-bottom: 0;
 padding-left: 23px;
 text-align: left;
 font-weight: 100;
}

#menu
{
 background-image: url(../images/menu.png);
 background-color: #6D6D6D;
 background-repeat: no-repeat;
 height: 32px;
 line-height: 34px;
 padding-left: 0px;
}

#menu li
{
 float: left;
 list-style-type: none;
}

#menu li a
{
 display: block;
```

```
 padding-top: 0;
 padding-right: 10px;
 padding-bottom: 0;
 padding-left: 10px;
 text-decoration: none;
 color: #fff;
}

#menu li a:hover
{
 color: #fff;
 text-decoration: none;
 background-image: url(../images/menuover.jpg);
 background-color: #577141;
 background-repeat: repeat-x;
}

#content
{
 background-color: #FFFFFF;
 padding-top: 10px;
 padding-right: 10px;
 padding-bottom: 10px;
 padding-left: 10px;
}

#left
{
/* background-color: rgb(12, 33, 141);*/
background-color: #FFFFFF;
 padding-top: 10px;
 padding-right: 10px;
 padding-bottom: 10px;
```

```
 padding-left: 10px;
 width: 210px;
 height: 410px;
 float: left;
 font-size: 12px;
 text-align: justify;
}

#left h3
{
/* color: rgb(76, 140, 206);*/
color: #555893;
 font-size: 18px ;
 border-bottom-color: rgb(76, 140, 206);
 border-bottom-width: 1px;
 border-bottom-style: dotted;
}

#right
{
/* background-color: rgb(203, 202, 255);*/
 padding-top: 20px;
 padding-right: 0;
 padding-bottom: 20px;
 padding-left: 30px;
 width: 630px;
 float: left;
 text-align: justify;
}

#right h3
{
 font-size: 17px;
```

```
 border-style: none;
 padding-top: 0;
 padding-right: 0;
 padding-bottom: 0;
 padding-left: 10px;
 margin: 0;
 color: #111;
 height: 30px;
 line-height: 30px;
}

#right ul
{
 list-style-type: none;
 padding-top: 10px;
 padding-right: 0;
 padding-bottom: 20px;
 padding-left: 20px;
}

#right ul li
{
 padding-top: 2px;
 padding-right: 0;
 padding-bottom: 3px;
 padding-left: 0;
}

#right ul li a
{
 color: #5C6F4C;
 font-weight: 100;
 display: block;
```

```
 text-decoration: none;
 font-size: 14px;
}

#right ul li a:hover
{
 color: #999;
}

#footer
{
 background-image: url(../images/Footer.png);
 background-color: #6D6D6D;
 background-repeat: repeat;
 font-size: 10px;
 color: #16f;
 text-align: center;
 height: 38px;
}

#footer a
{
 color: red;
 text-decoration: none;
}

#footer a:hover
{
 color: #aaa;
 text-decoration: underline;
}

#logo
```

```
{
 background-image: url("../images/logo.jpg");
 background-repeat: no-repeat;
 height: 95px;
 width: 480px;
}

#logo2
{
 background-image: url("../images/logo.jpg");
 background-repeat: no-repeat;
 height: 95px;
 width: 480px;
}

#cabecera
{
 background-image: url("../images/header.png");
 background-repeat: no-repeat;
 height: 40px;
 left: 0;
 top: 0;
 width: 944px;
 z-index: -2;
}

/*fieldset*/

fieldset{
    font-family: sans-serif;
    color: #555893;
    font-weight: bold;
    font-size: 10pt;
```

```
 }


 /*formularios*/
 form label{
     font-family: sans-serif;
     color: #0B0B61;
     font-weight: bold;
     font-size: 8pt;
 }
 /*ingresar solo mayusculas en el form*/
 input.mayusculas{text-transform:uppercase;}
/*botones*/
a.button2{
    background:url("../images/botones/boton.png");
    display:block;
    color: #0B0B61;
    font-weight:bold;
    height:30px;
    line-height:29px;
    margin-bottom:14px;
    text-decoration:none;
    width:191px;
}
a:hover.button2{
color:#0066CC;
}

/* -------------------- */
/* CLASSES */
/* -------------------- */
.add{
    background:url("../images/botones/add.ico") no-repeat 10px 8px;
```

```
    text-indent:30px;
    display:block;
}
.delete{
    background:url("../images/botones/delete.ico") no-repeat 10px 8px;
    text-indent:30px;
    display:block;
}
.cancel{
    background:url("../images/botones/cancel.ico") no-repeat 10px 8px;
    text-indent:30px;
    display:block;
}
.update{
    background:url("../images/botones/update.ico") no-repeat 10px 8px;
    text-indent:30px;
    display:block;
}
```

Con esto tenemos un mayor control de los estilos de todos los componentes del formulario dando una imagen personalizada y agradable para el usuario acorde a la plantilla que se utiliza para el proyecto.

1.1.6 CONEXIÓN CON BASE DE DATOS

Ahora podemos definir las reglas de negocio en las que está basada nuestra aplicación. Estas reglas de negocio son lo que constituyen el Modelo, y estás están basadas en los datos que se encuentran en nuestra base de datos.

La base de datos utilizada está íntegramente realizada en Postgres teniendo acceso a nuestra base de datos de la aplicación a través del archivo de configuración application.ini, siendo la siguiente:

```
resources.db.adapter = PDO_PGSQL
resources.db.params.host = localhost
resources.db.params.username = majo
resources.db.params.password = admin
resources.db.params.dbname = sgmas
```

Zend Framework facilita la clase Zend_Db_Table la cual permite acceder a los datos que se encuentra en la Base de Datos. Para proyectos de mayor envergadura, se suelen crear una o más instancias de la clase Zend_Db_Table.

Zend_Db_Table es una clase abstracta, por lo que tenemos que crear una que herede de ella. El nombre que tendrá, vendrá definida por el prefijo Model_DbTable_ y el nombre de la tabla con la que se va a interactuar, nombre que además informaremos en la variable protegida $_name.

Zend_Db_Table asume que nuestra tabla tiene una clave primaria que se llama id y que este es un auto incremental. El nombre de la clave primaria se puede modificar si así se desea.

Nuestra nueva clase la crearemos en el archivo Producto.php, dentro de /applications/models/DbTable/producto.php, desde este archivo podemos controlar todos los datos guardados en la base en forma de métodos que mediante instancias podemos acceder a las funciones que se encuentran en el mismo. De esta forma tenemos conformado el archivo Producto.php que maneja el modelo de la base de datos:

```
<?php
```

```
class Application_Model_DbTable_Producto extends
Zend_Db_Table_Abstract
{

    protected $_name = 'tb_producto';
    public function get($id_producto)
    {
        $id_producto = (int) $id_producto;
        //$this->fetchRow devuelve fila donde id = $id
        $row = $this->fetchRow('id_producto = ' . $id_producto);
        if (!$row)
        {
            throw new Exception("No se puede encontrar el registro
            $id_producto");
        }
        return $row->toArray();
    }

        public function get1($id_producto)
    {
        $id_producto = (int) $id_producto;
        //$this->fetchRow devuelve fila donde id = $id
        return $this->fetchRow('id_producto = ' . $id_producto);
    }


    /**
     *  agrega un nuevo producto a la base de datos
     */
    public function agregar($id_bodega, $id_familia, $id_presentacion,
        $id_unidad, $id_comercial ,$nombre_producto, $estado_producto,
        $observacion_producto, $stock_max, $stock_min)
    {
```

```
        $data = array('id_bodega' => $id_bodega, 'id_familia' => $id_familia,
  'id_presentacion'=> $id_presentacion,
        'id_unidad' => $id_unidad, 'id_comercial' => $id_comercial,
'nombre_producto' => $nombre_producto, 'estado_producto' =>
$estado_producto,
        'observacion_producto' => $observacion_producto, 'stock_max' =>
$stock_max, 'stock_min' => $stock_min);
        //$this->insert inserta nuevo album
        $this->insert($data);
    }

    public function cambiar($id_producto,$id_bodega, $id_familia,
    $id_presentacion, $id_unidad, $id_comercial,$nombre_producto,
    $estado_producto, $observacion_producto, $stock_max, $stock_min)
    {
        $data = array('id_producto' => $id_producto, 'id_bodega' =>
        $id_bodega, 'id_familia' => $id_familia, 'id_presentacion'=>
        $id_presentacion, 'id_unidad' => $id_unidad, 'id_comercial' =>
        $id_comercial, 'nombre_producto' => $nombre_producto,
        'estado_producto' => $estado_producto,
        'observacion_producto' => $observacion_producto, 'stock_max' =>
        $stock_max, 'stock_min' => $stock_min);
        //$this->update cambia datos del producto con id= $id
        $this->update($data, 'id_producto = ' . (int) $id_producto);
    }

    public function borrar($id)
    {
        //$this->delete borrar album donde id=$id
        $this->delete('id_producto =' . (int) $id);
    }

    public function listar()
```

```
    {
        //devuelve todos los registros de la tabla
        return $this->fetchAll();
    }

         public function listar_reporte()
    {

       $db = Zend_Registry::get('pgdb');
       $db->setFetchMode(Zend_Db::FETCH_OBJ);
       //hago la consulta sql que desee:
       //
        //devuelve todos los registros de la tabla
        return $db->fetchAll("SELECT P.ID_PRODUCTO,
        B.NOMBRE_BODEGA, F.NOMBRE_FAMILIA,
        PR.NOMBRE_PRESENTACION, U.UNIDAD_MEDIDA,
        P.NOMBRE_PRODUCTO, P.ESTADO_PRODUCTO,
        P.OBSERVACION_PRODUCTO, P.STOCK_MAX, P.STOCK_MIN
        FROM TB_PRODUCTO P, TB_BODEGA B, TB_FAMILIA F,
        TB_PRESENTACION PR, TB_UNIDADES U
        WHERE P.ID_BODEGA=B.ID_BODEGA AND
       P.ID_FAMILIA=F.ID_FAMILIA AND
       P.ID_PRESENTACION=PR.ID_PRESENTACION AND
       P.ID_UNIDAD=U.ID_UNIDAD");
    }

   public function buscar_producto1($consulta)
    {
        //recupero objeto desde el registro
        $db = Zend_Registry::get('pgdb');
        //opcional, esto es para que devuelva los resultados como objetos
$row->campo
        $db->setFetchMode(Zend_Db::FETCH_OBJ);
```

```
        return $db->fetchRow("SELECT ESTADO_PRODUCTO
                              FROM TB_PRODUCTO
                              WHERE NOMBRE_PRODUCTO LIKE
'%".$consulta."%'");
    }

         public function consulta_produc($consulta)
    {

       $db = Zend_Registry::get('pgdb');
       //opcional, esto es para que devuelva los resultados como objetos $row-
>campo
       $db->setFetchMode(Zend_Db::FETCH_OBJ);
       //hago la consulta sql que desee:
              $select="SELECT *
               FROM tb_producto AS p
               INNER JOIN tb_bodega AS b ON (p.id_bodega = b.id_bodega)
               INNER JOIN tb_familia AS f ON (p.id_familia = f.id_familia)
               INNER JOIN tb_presentacion AS pr ON (p.id_presentacion =
pr.id_presentacion)
               INNER JOIN tb_unidades AS u ON (p.id_unidad = u.id_unidad)
               WHERE (p.nombre_producto LIKE '%".$consulta."%')
               OR (p.observacion_producto LIKE '%".$consulta."%')";

       return $db->fetchAll($select);
    }

         public function consulta_producto($parametro)
    {

       $db = Zend_Registry::get('pgdb');
```

```
        //opcional, esto es para que devuelva los resultados como objetos $row->campo
        $db->setFetchMode(Zend_Db::FETCH_OBJ);
        //hago la consulta sql que desee:
        return $db->fetchAll("SELECT * FROM TB_PRODUCTO AS p
                        INNER JOIN tb_presentacion AS pr ON (p.id_presentacion = pr.id_presentacion)
                        INNER JOIN tb_unidades AS u ON (p.id_unidad = u.id_unidad)
                        WHERE (p.NOMBRE_PRODUCTO LIKE '%".$parametro."%')
                          OR (p.OBSERVACION_PRODUCTO LIKE '%".$parametro."%')");
    }
}
```

1.2 DICCIONARIO DE DATOS.

El diccionario de datos contiene las características lógicas de los datos que se van a utilizar en el Módulo de Gestión de Medicamentos del Sistema de Gestión Médico para Áreas de Salud – SGMAS, de los datos que se desarrollan durante el análisis de flujo de datos y los requerimientos del sistema.

Por lo general, el diccionario de datos está integrado en el sistema de información el cual detallamos el contenido cada tabla con sus respectivos campos, tipos, relaciones de la base de datos que nos sirvió para el desarrollo del módulo.

Tabla tb_bajas

BAJAS			
NOMBRE DE LA TABLA		tb_bajas	
CAMPO	**TIPO**	**RELACIÓN**	**DATOS**
id_lote	INT8 (pk,fk)	tb_lote - tb_bajas	Código lote
id_producto	INT8 (pk,fk)	tb_producto - tb_bajas	Código producto
cantidad_baja	INT8		Cantidad de medicamento dados de baja
fecha_baja	DATE		Fecha de ingreso de la baja
razon_baja	VARCHAR(100)		Motivo de la baja

Tabla tb_cantidad_lote

EXISTENCIAS			
NOMBRE DE LA TABLA		tb_cantidad_lote	
CAMPO	**TIPO**	**RELACIÓN**	**DATOS**
id_lote	VARCHAR(20) (pk, fk)	tb_lote - tb_cantidad_lote	Código lote
id_producto	INT8 (pk, fk)	tb_producto - tb_cantidad_lote	Código producto
cantidad_lote	INT8		Cantidad de medicamento existente en bodega

Tabla tb_bodega

BODEGA			
NOMBRE DE LA TABLA		tb_bodega	
CAMPO	**TIPO**	**RELACIÓN**	**DATOS**
id_bodega	INT8 (pk)		Código de bodega *
nombre_bodega	VARCHAR(50)		Nombre de la bodega donde se almacena el medicamento
direccion_bodega	VARCHAR(100)		Dirección de la bodega donde se almacena el medicamento
telefono_bodega	VARCHAR(20)		Teléfono de la bodega donde se almacena el medicamento

Tabla tb_documento

DETALLE DOCUMENTOS CONTABLES			
NOMBRE DE LA TABLA		tb_documento	
CAMPO	**TIPO**	**RELACION**	**DATOS**
id_doc	INT8 (pk)		Código del documento contable*
ruc_prov	INT8 (fk)	tb_proveedores - tb_documento	Código del proveedor
id_tipo_doc	INT8 (fk)	tb_tipo_doc - tb_documento	Código del tipo de documento contable (factura, nota de venta, etc.)
fecha_doc	DATE		Fecha de emisión del documento contable
valor_doc	DECIMAL(8,0)		Monto total del documento contable
num_ref_doc	VARCHAR(20)		Número de referencia del documento contable

Tabla tb_familia

FAMILIA			
NOMBRE DE LA TABLA		tb_familia	
CAMPO	**TIPO**	**RELACIÓN**	**DATOS**
id_familia	INT8 (pk)		Código de la familia*
nombre_familia	VARCHAR(50)		Nombre de la familia dependiendo del almacenamiento en bodega (medicamentos, insumos médicos, etc.)
observaciones_familia	VARCHAR(100)		Observaciones referentes al caso

Tabla tb_movimiento

MOVIMIENTOS			
NOMBRE DE LA TABLA		tb_movimiento	
CAMPO	**TIPO**	**RELACIÓN**	**DATOS**
id_movimiento	INT8 (pk)		Código del movimiento*
id_bodega	INT8 (fk)	tb_bodega - tb_movimiento	Código de la bodega donde se realiza el movimiento
id_transaccion	INT8 (fk)	tb_transaccion - tb_movimiento	Código de transacción
responsable_movimiento	VARCHAR(50)		Responsable de la realización del movimiento
fecha_movimiento	DATE		Fecha de realización del movimiento
estado_movimiento	VARCHAR(20)		Se almacena el estado del movimiento dependiendo del pedido (despachado, pendiente, negado)
tipo_trans_movimiento	VARCHAR(50)		Tipo de transacción dependiendo del código de transacción al que pertenece (egreso de bodega, pedido, etc.)
tipo_pedido	VARCHAR(50)		

Tabla tb_mov_detalle

DETALLE DEL MOVIMIENTO			
NOMBRE DE LA TABLA		tb_mov_detalle	
CAMPO	**TIPO**	**RELACIÓN**	**DATOS**
id_movimiento	INT8 (pk, fk)	tb_movimiento - tb_mov_detalle	Código del movimiento
id_lote	VARCHAR(20) (pk, fk)	tb_producto - tb_mov_detalle	Código del producto
id_producto	INT8 (pk, fk)	tb_lote - tb_mov_detalle	Codigo del lote
cantidad_mov_prod	INT8		Cantidad de medicación solicitada en el movimiento

Tabla tb_lote

LOTE DE MEDICAMENTOS			
NOMBRE DE LA TABLA		tb_lote	
CAMPO	**TIPO**	**RELACION**	**DATOS**
id_lote	VARCHAR(20) (pk)		Código del lote de medicación*
id_producto	INT8 (pk,fk)	tb_producto - tb_lote	Código producto
id_doc	INT8	tb_doc - tb_lote	Código de documento contable
marca_lote	VARCHAR(20)		Marca o empresa encargada de su fabricación
descripcion_lote	VARCHAR(50)		Breve descripción del lote
recepción_lote	DATE		Fecha de recepción del lote de medicación
elaboración_lote	DATE		Fecha de elaboración del lote de medicación
caducidad_lote	DATE		Fecha de caducidad del lote de medicación
costo_lote	DECIMAL(8,0)		Costo por lote de medicación
cantidad_lote	INT8		Cantidad recibida del medicamento

Tabla tb_presentacion

PRESENTACION			
NOMBRE DE LA TABLA		tb_presentacion	
CAMPO	**TIPO**	**RELACION**	**DATOS**
id_presentacion	INT8 (fk)		Código de la presentación del medicamento*
nombre_presentacion	VARCHAR(50)		Nombre de la presentación del medicamento a almacenar (envases, tabletas, frasco, spray, etc.)
observaciones_presentacion	VARCHAR(100)		Observaciones referentes al caso

Tabla tb_producto

MEDICAMENTO			
NOMBRE DE LA TABLA		tb_producto	
CAMPO	**TIPO**	**RELACION**	**DATOS**
id_producto	INT8 (pk)		Código producto*
id_bodega	INT8 (fk)	tb_bodega-tb_producto	Código de la bodega donde se encuentra almacenado el medicamento
id_familia	INT8 (fk)	tb_familia - tb_producto	Código de la familia de la que pertenece le producto
id_presentacion	INT8 (fk)	tb_presentacion - tb_producto	Código de la presentación que posee la medicación
id_unidad	INT8 (fk)	tb_unidad - tb_producto	Código de la unidad de medida del producto
nombre_producto	VARCHAR(100)		Nombre del medicamento
estado_producto	VARCHAR(20)		Dependiendo del medicamento este puede ser: rezagado, descontinuado, etc.
observación_producto	VARCHAR(100)		Se colocara alguna observación a la medicación ingresada
stock_max	INT8		Campo de control de exceso de stock
stock_min	INT8		Campo de control de déficit de stock

Tabla tb_proveedores

PROVEEDORES			
NOMBRE DE LA TABLA		tb_proveedores	
CAMPO	**TIPO**	**RELACION**	**DATOS**
ruc_prov	INT8 (pk)		RUC o número de cédula del proveedor
nombre_prov	VARCHAR(50)		Nombre del proveedor que distribuye o entrega el medicamento
direccion_prov	VARCHAR(100)		Dirección del proveedor que distribuye o entrega el medicamento
telefono_prov	VARCHAR(20)		Teléfono del proveedor que distribuye o entrega el medicamento
responsable_prov	VARCHAR(50)		Campo para identificar el responsable de la entrega de la medicación

Tabla tb_responsable

RESPONSABLES			
NOMBRE DE LA TABLA		tb_responsable	
CAMPO	**TIPO**	**RELACION**	**DATOS**
id_responsable	INT8 (pk)		Código responsable*
id_bodega	INT8 (fk)	tb_bodega- tb_responsable	Código de la bodega que identifica el responsable de la misma
nombre_responsable	VARCHAR(50)		Nombre del responsable o encargado de la bodega
direccion_responsable	VARCHAR(100)		Dirección del responsable o encargado de la bodega
telefono_responsable	VARCHAR(20)		Teléfono del responsable o encargado de la bodega
mail_responsable	VARCHAR(20)		Mail del responsable o encargado de la bodega

Tabla tb_tipo_doc

TIPO DE DOCUMENTO CONTABLE			
NOMBRE DE LA TABLA		tb_tipo_doc	
CAMPO	**TIPO**	**RELACION**	**DATOS**
id_tipo_doc	INT8 (pk)		Código tipo de documento contable*
tipo_doc	VARCHAR(50)		Tipo de documento contable (nota de venta, factura, recibo, etc.)

Tabla tb_unidades

UNIDAD DE MEDIDA DEL MEDICAMENTO			
NOMBRE DE LA TABLA		tb_unidades	
CAMPO	**TIPO**	**RELACION**	**DATOS**
id_unidad	INT8 (pk)		Código de la unidad de medida*
unidad_medida	VARCHAR(50)		Unidad de medida dependiendo al medicamento
observaciones_unidad	VARCHAR(50)		Observaciones referentes al caso

UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO-CAMPUS SUR

CARRERA DE INGENIERÍA DE SISTEMAS

MENCIÓN TELEMÁTICA

ANÁLISIS, DISEÑO Y DESARROLLO DE UN MÓDULO PARA LA GESTIÓN DE MEDICAMENTOS DEL SISTEMA DE GESTIÓN MÉDICO PARA ÁREAS DE SALUD (SGMAS), PARA EL CENTRO DE SALUD NO. 3 "LA TOLA-VICENTINA" DE LA DIRECCIÓN PROVINCIAL DE SALUD DE PICHINCHA.

GUÍA DEL USUARIO

CARLOS FERNANDO ARIAS MONTALVO
MARÍA JOSÉ MICHELENA PERUGACHI

DIRECTOR ING. RENÉ ARÉVALO

Quito, Octubre del 2012

GUIA DEL USUARIO

TABLA DE CONTENIDOS

INICIO DEL PROGRAMA

1. ACCESO DIRECTO

El acceso directo de la Aplicación nos lleva a la siguiente dirección http://localhost/sgmas-med/public/index que se encuentra ubicado en el escritorio.

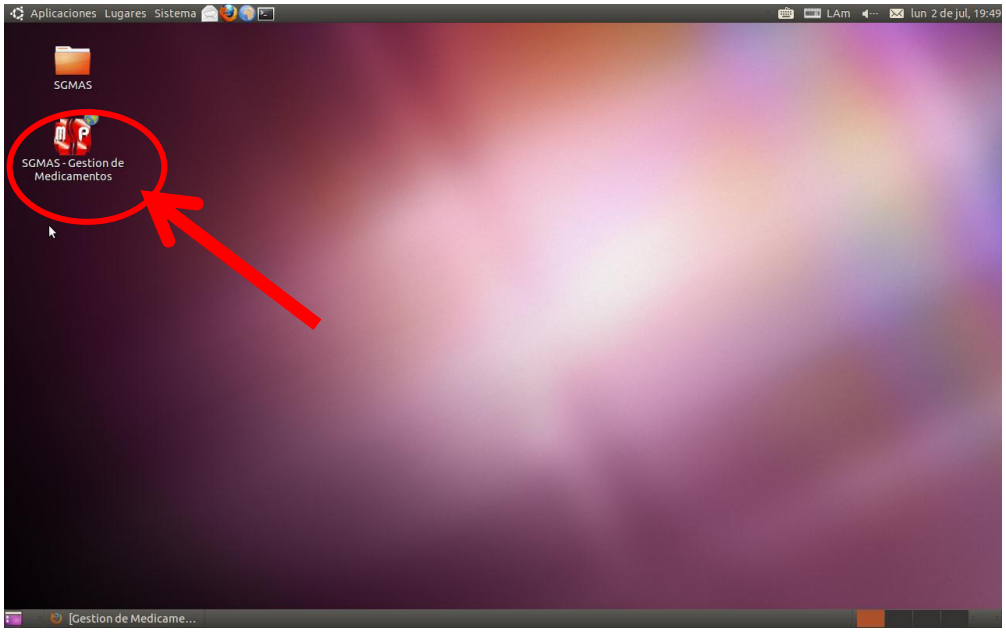


Figura 1. Acceso al Módulo Gestión de Medicamentos. [92]

2. PANTALLA DE INICIO

Al acceder a la aplicación procedemos a indicar el nombre de usuario y contraseña otorgado por el administrador del Sistema el cual será verificado para su confirmación de ingreso de lo contrario se mostrará un mensaje de error para que se revise los datos ingresados y se los pueda corregir.

[92] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

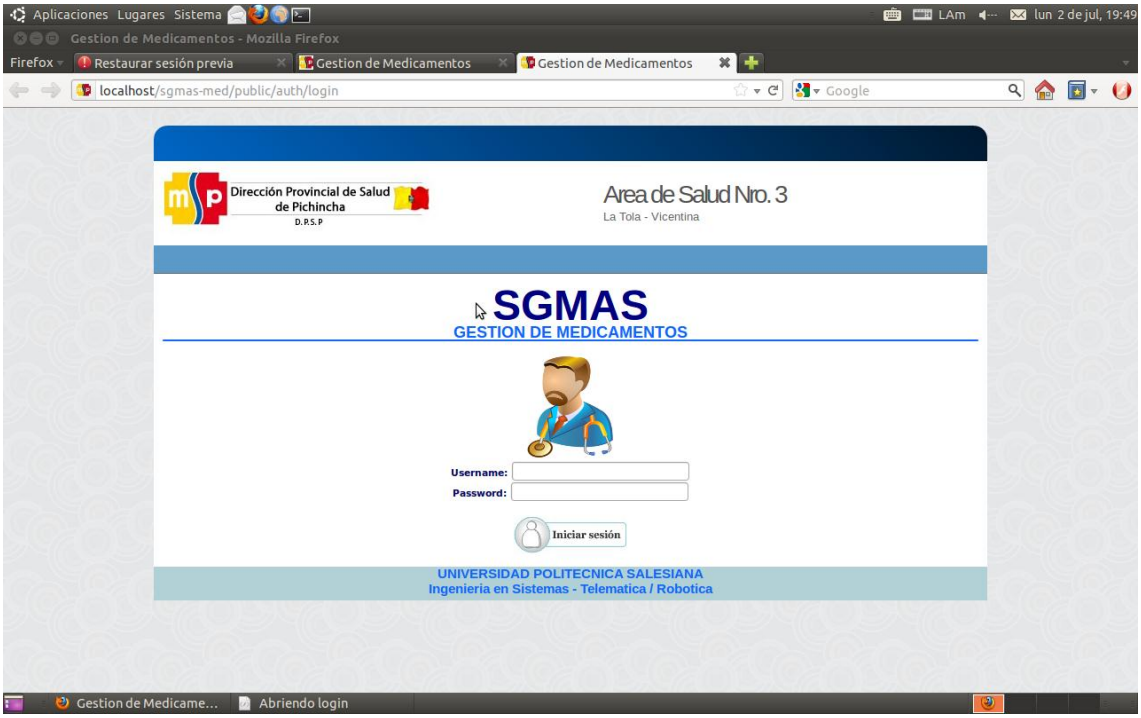


Figura 2.Pantalla de Inicio. [93]

3. PANTALLA PRINCIPAL

Dadas las respectivas verificaciones y otorgados los permisos para el acceso a la aplicación esta nos dirige a la pantalla principal.

[93] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

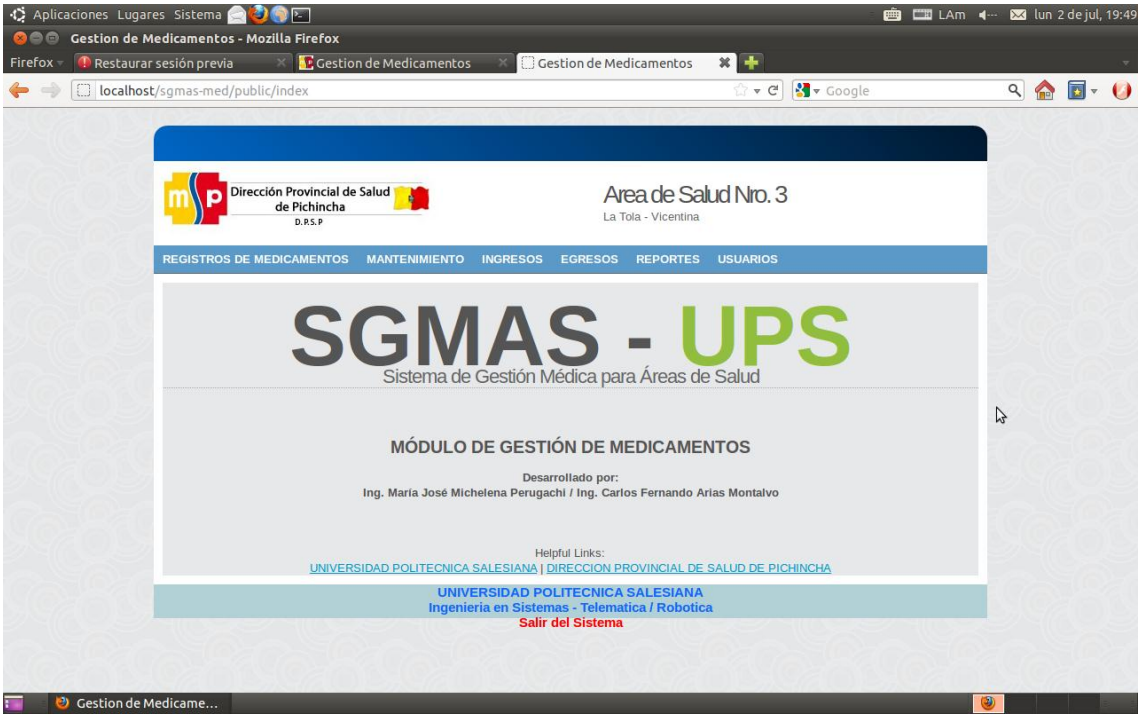


Figura 3. Pantalla Principal. [94]

En esta se encuentra todo el menú de la que se compone el Sistema de Gestión de Medicamentos que en su forma consta de:

- **REGISTRO DE MEDICAMENTOS:**

  Donde se encuentran las interfaces que se dedican de la administración de la información y recursos de administración de los productos; y son:

  - ***Productos:*** Interfaz que maneja los ingresos de la información general del medicamentos a ingresarlo cabe destacar que esta información se la llena una sola vez en el sistema y que solo se puede modificar dependiendo del mantenimiento que el administrador de a la aplicación.

[94] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

- ***Ubicación Física:*** Interfaz que provee al usuario del Sistema tener un control en cuanto al espacio físico y la ubicación que el usuario otorgue al lote de medicación ingresado esto permitirá realizar búsquedas físicas y organización eficiente.

- ***Registro de Bajas:*** Interfaz que permite de una manera eficiente al administrador del Sistema manejar las bajas que el medicamento pueda tener por causa de cualquier siniestro (terremotos, incendios, inundaciones, etc.) que necesite ser registrado.

  Este registro lo realiza únicamente el administrador del Sistema es una de las interfaces restringidas por los datos y los productos que se manejan.

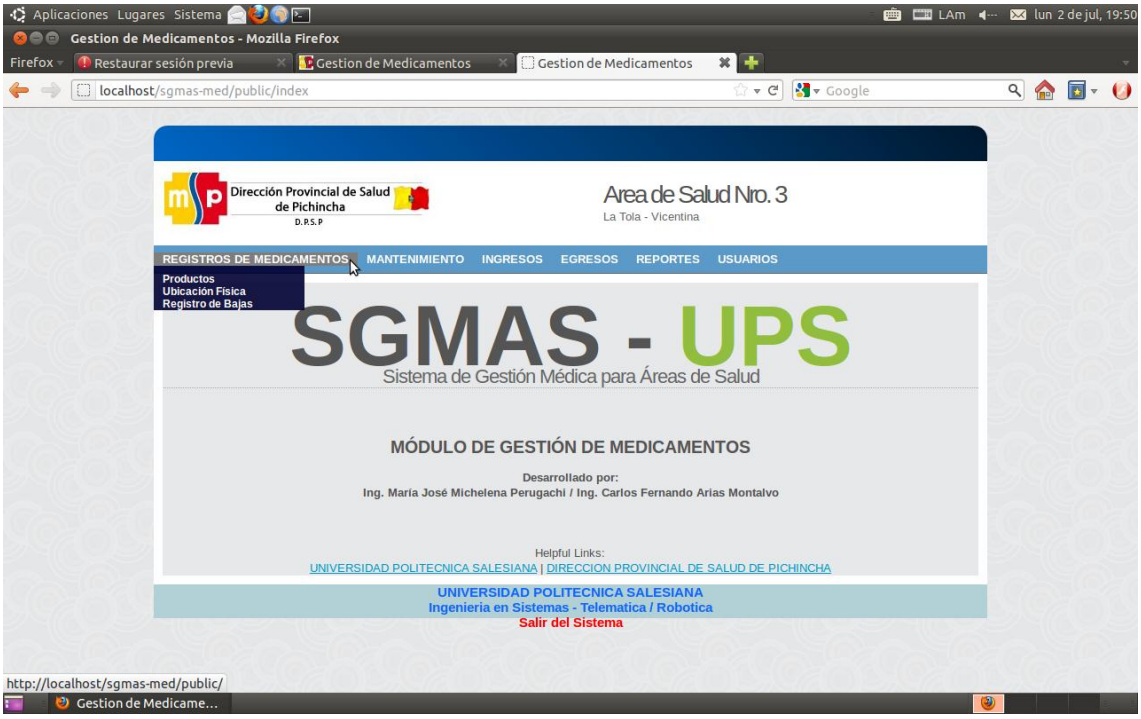


Figura 4.Registro de Medicamentos. [95]

[95] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

4. PRODUCTOS

Se ingresan de una manera ordenada los campos generales que permiten identificar de una manera al medicamento a posteriormente ser ingresado a cada lote de medicación.

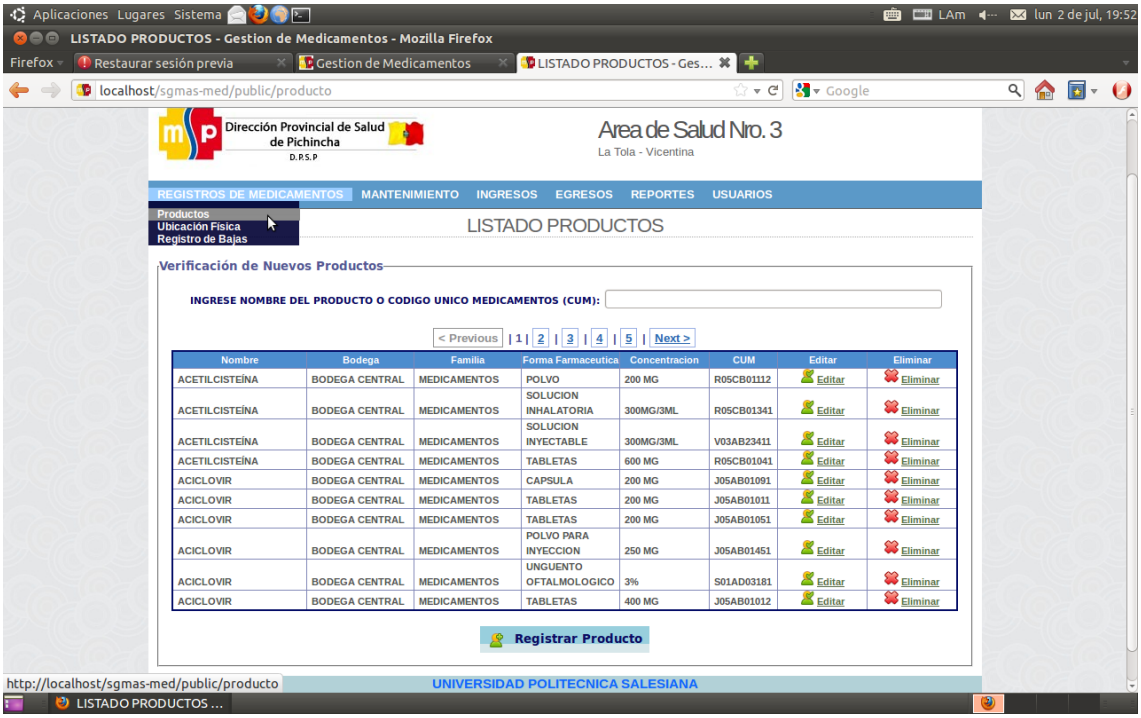


Figura 5. Listado de Productos [96]

Se pueden manejar elementos de ingreso, edición y eliminación de campos y datos ingresados.

Se muestra una pantalla de inicio en el mismo se encuentra una tabla que muestra campos con datos anteriormente ingresados que se los puede verificar mediante un filtro de búsqueda incremental que muestra la existencia o no del producto.

Al verificar que no se tiene registrado el medicamento se lo procede a ingresar pulsando “REGISTRAR PRODUCTO”, botón que se encuentra

[96] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

muy bien identificado en la parte inferior centrada de en la pantalla de inicio.

A continuación mostraremos los campos que de una manera visual se indica la obligatoriedad del ingreso de los mismos.

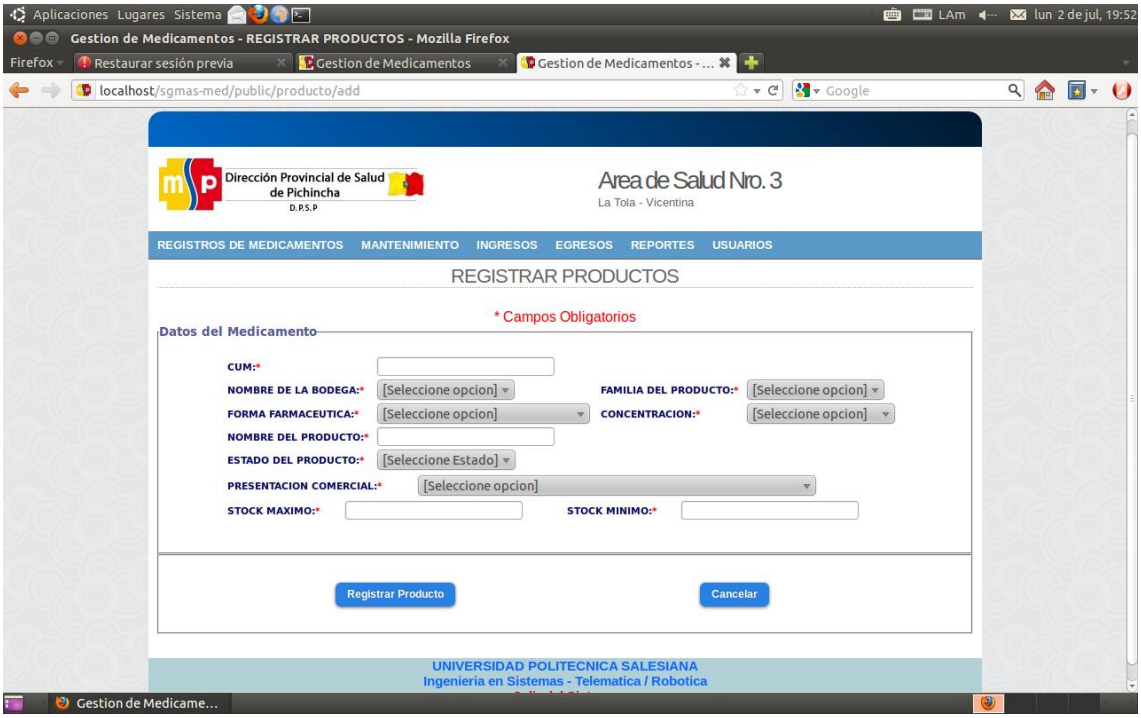


Figura 6. Registrar Productos. [97]

Cabe destacar que se requieren datos obligatorios de las tablas de mantenimiento que se encuentran en la administración de catálogos que explicaremos a continuación:

- **Nombre de la Bodega**

  Nombre de la bodega en donde se hace el registro.

[97] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

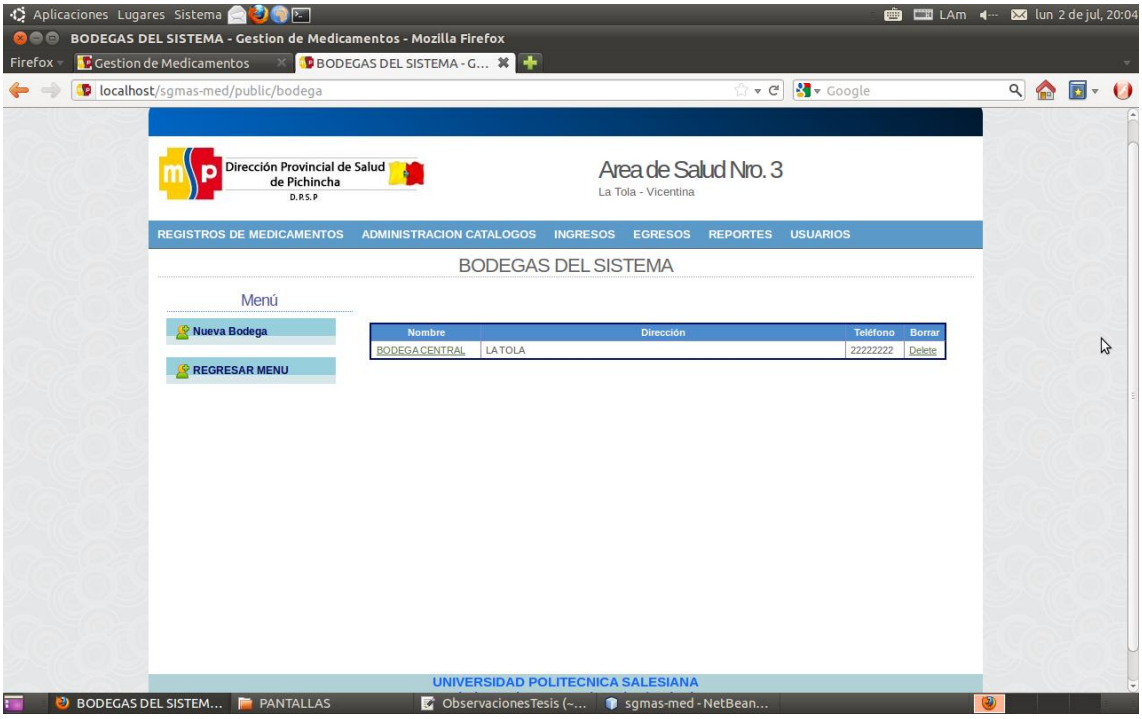


Figura 7.Bodegas del Sistema. [98]

- **Responsable de Bodega**

Después de ingresada la bodega se procede a asignarle el responsable de esa bodega es por eso que el nombre de la persona debe estar registrada en la tabla de responsables que así mismo se encuentra en el menú de administración de catálogos.

Los datos que se almacenan en esta tabla es información general de la persona responsable de cada una de las bodegas en las que se va a recibir la medicación como método de control de los lotes de la medicación que ingresa o se despacha.

[98] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

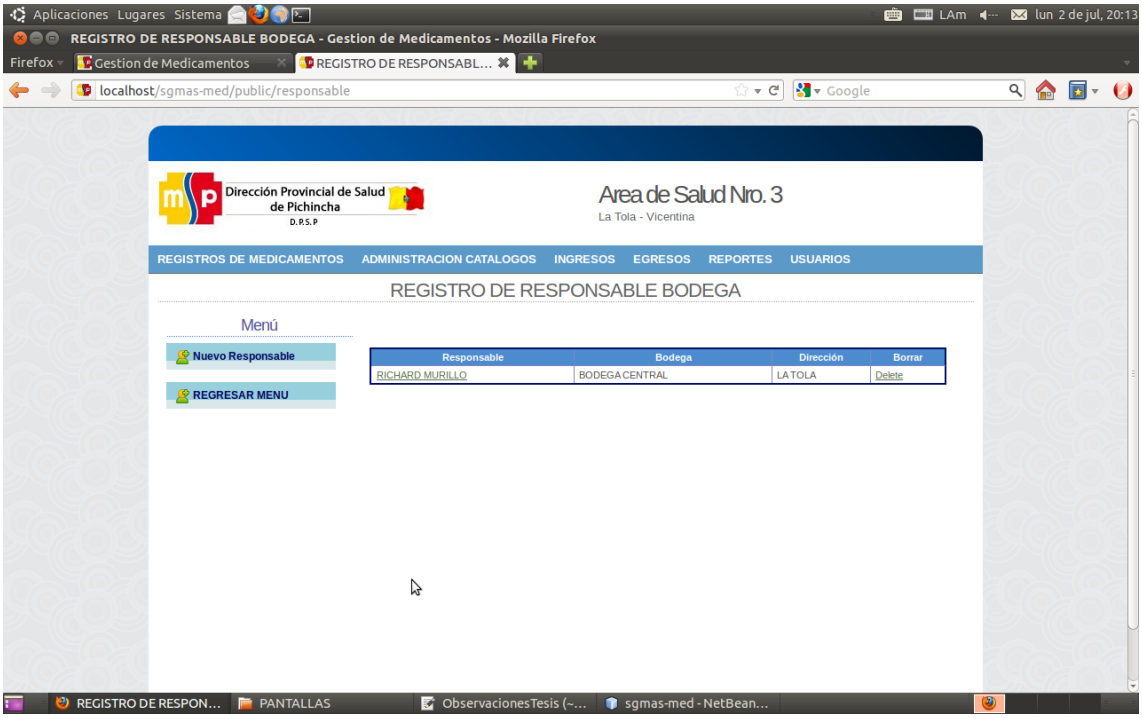


Figura 8. Registro de Responsable de Bodega. [99]

- **Familia del Producto**

  Indica si el producto es un medicamento o cualquier otro insumo médico. Para la elaboración de la aplicación que se dedica exclusivamente a la administración de medicamentos se ha tomado en cuenta solo los productos médicos de consumo y no de insumos pero dado la naturaleza del Sistema y de las herramientas en el que se encuentra elaborado esta puede ser fácilmente adaptable a las necesidades que la institución requiera.

[99] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

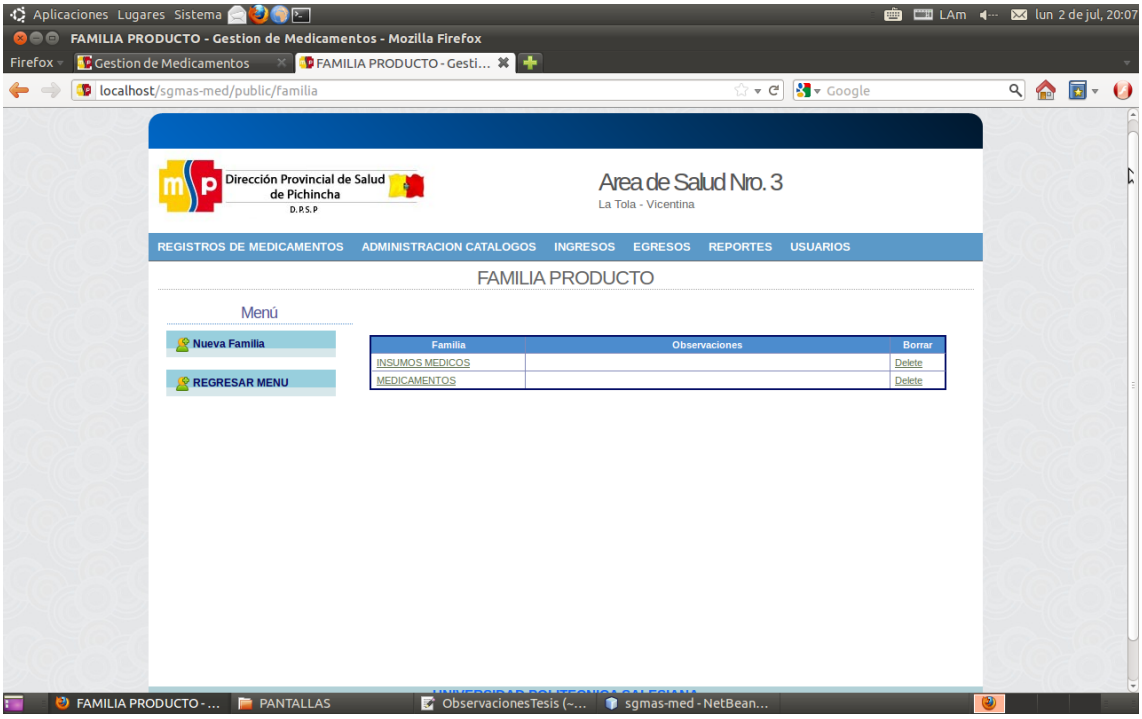


Figura 9. Familia del Producto. [100]

- **Forma Farmacéutica**

  Indica la presentación en el que el lote de medicación ha ingresado a la bodega ejemplo: polvo, solución inyectable, cápsula, etc.

[100] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

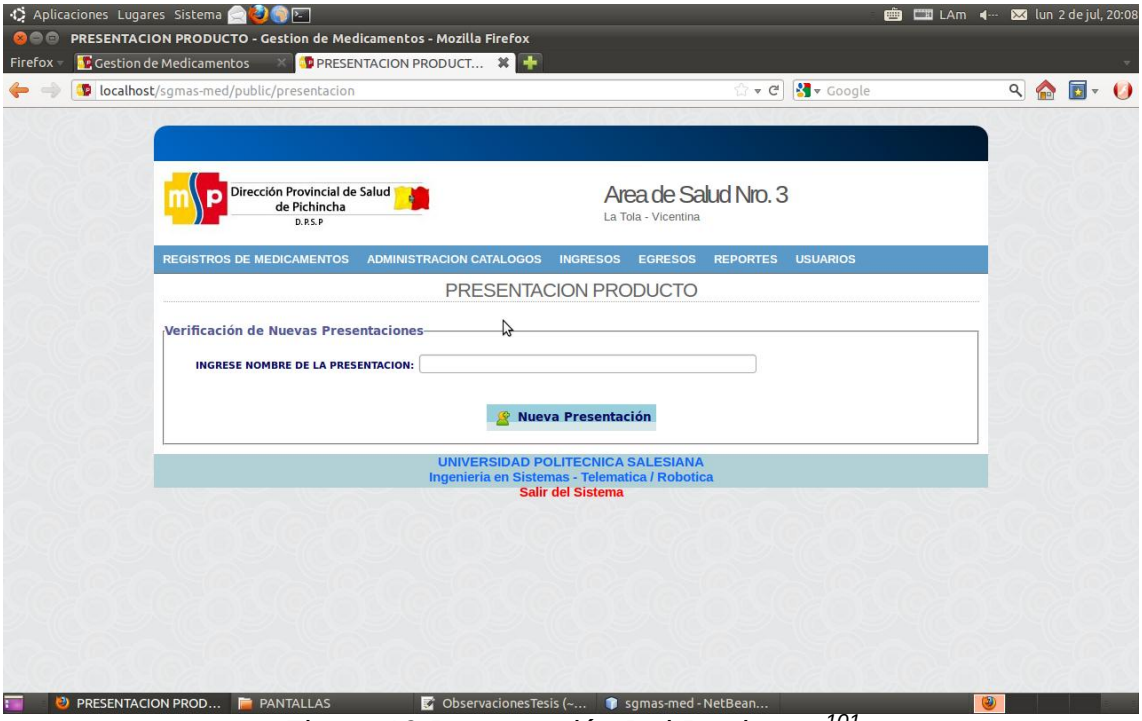


Figura 10.Presentación Del Producto. [101]

- **Concentración**

  Como su nombre lo indica muestra la concentración farmacéutica que tiene el medicamento; ejemplo: 100ml, 10g, etc.

[101] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

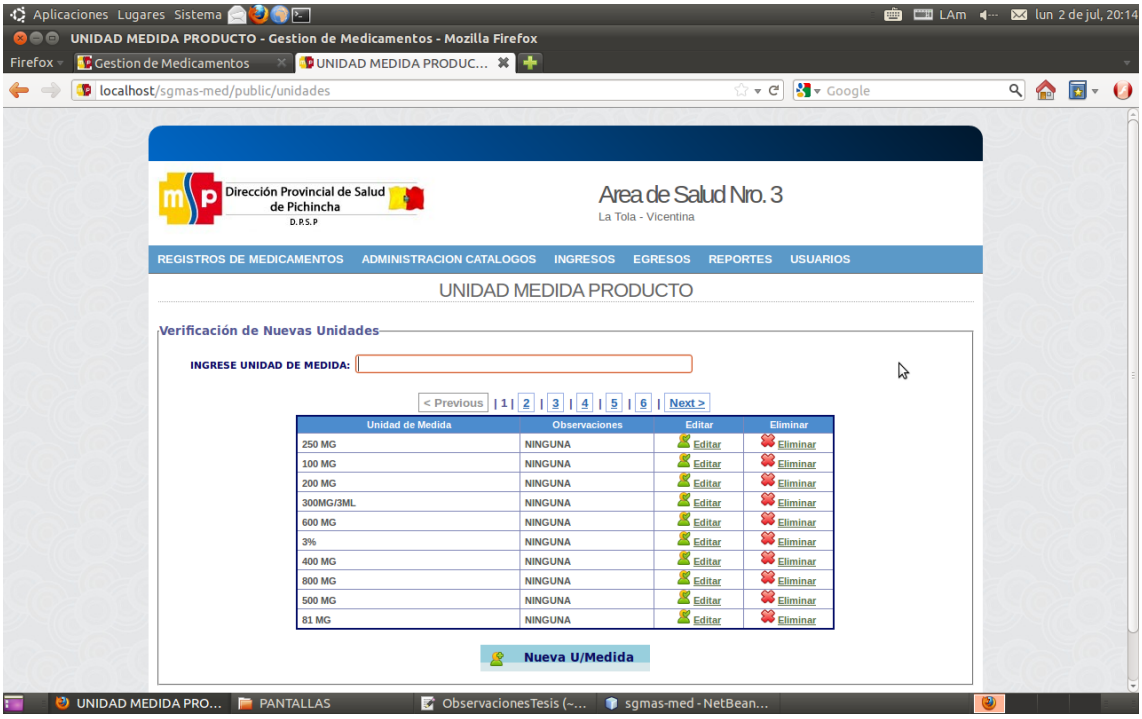


Figura 11. Unidad de Medida del Producto. [102]

- **Presentación Comercial**

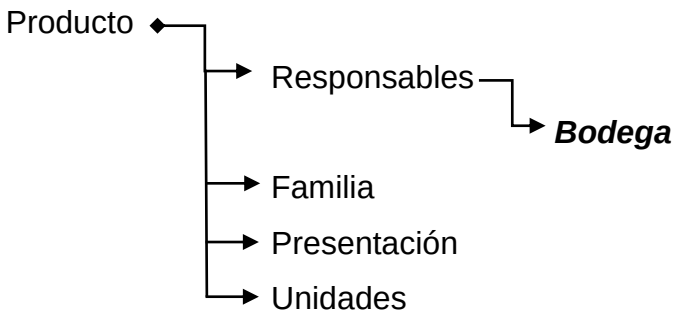
Indica la presentación en la que el producto o medicamento ingresado llega a la bodega por ejemplo: caja con un blíster de 10 unidades.

De no encontrarse los datos deseados para el ingreso de un registro se deberá obligatoriamente realizar su ingreso desde las “ADMINISTRACION DE CATALOGOS” con su respectiva tabla de mantenimiento para poder realizar el ingreso de los registros.

Todas interfaces que posee las el menú principal son de fácil utilización para el ingreso como para la edición y eliminación de los registros.

Para poder ingresar un nuevo medicamento se debe primero verificar los siguientes datos existentes:

[102] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*



5. INGRESOS

En la interfaz de ingresos y teniendo ingresado los datos generales del producto y/o medicación a registrar procedemos a ingresar al Sistema los LOTES DE MEDICACION que ingresan a la bodega de la misma forma se presenta una pantalla con datos específicos de la entrega y registro de cada lote de medicación así como datos de control del Sistema.

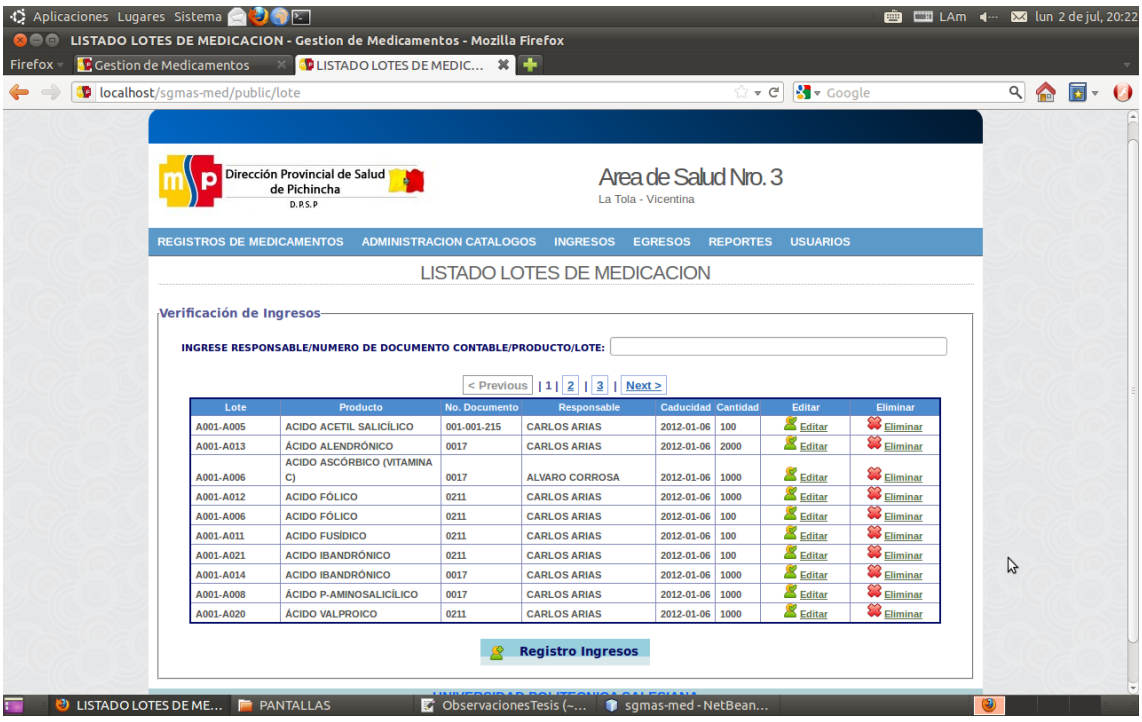


Figura 12. Listado de Lotes de Medicación. [103]

[103] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Al igual que se maneja la interfaz del ingreso de los datos generales de un medicamento, en el ingreso de un lote de medicación tenemos un filtro de búsqueda que nos permitirá constatar de manera rápida mediante una búsqueda secuencial los ingresos que se han realizado.

La búsqueda mediante este filtro se lo puede realizar por el nombre del responsable de la bodega, el nombre del producto y el número del documento contable con la que ingreso el lote de medicación.

Se procede al registro de un nuevo ingreso en la url http://localhost/sgmas-med/public/lote/add/ en la parte inferior central de la pantalla de inicio de ingresos que a continuación dirige a la siguiente interfaz:

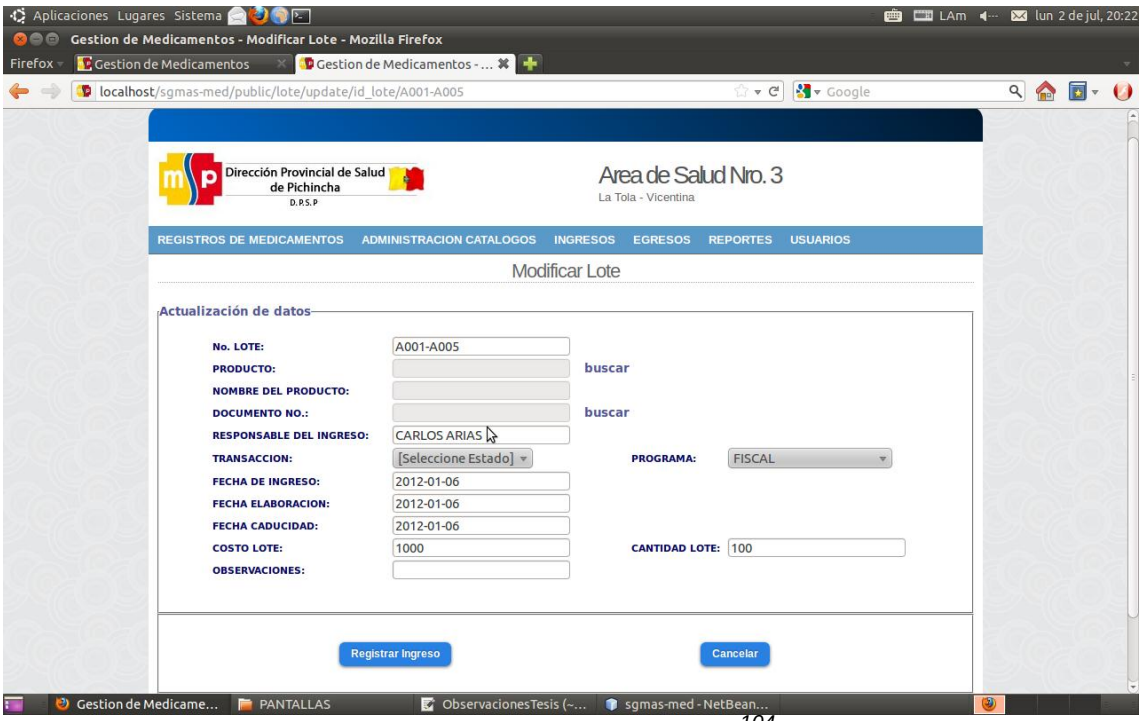


Figura 13. Modificar Lote. [104]

[104] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

En la interfaz se encuentran claramente identificadas los campos que deberían ser obligatorios y que deben encontrarse la mayoría guardados en las tablas de mantenimiento de administración de catálogos así como una interfaz de ayuda para búsqueda de los productos a ingresar para de esta forma cargar los datos al interfaz de ingreso de la misma manera funciona para la búsqueda y carga del numero de documento contable con el cual ingresa el lote de medicación.

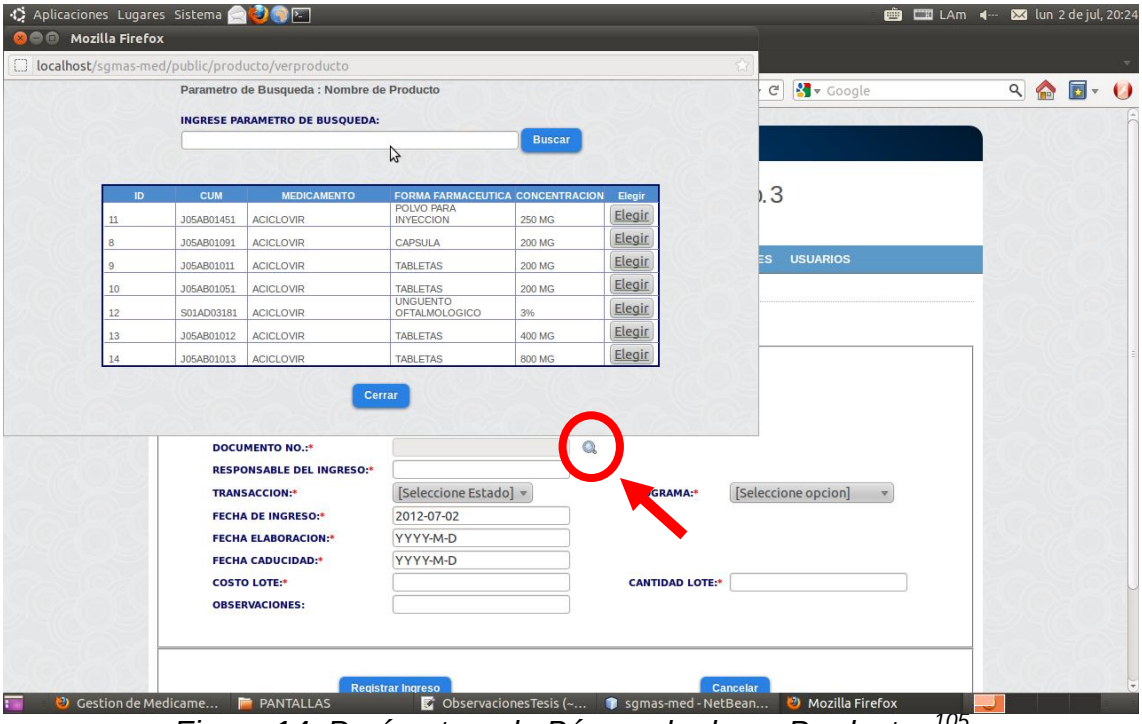


Figura 14. Parámetros de Búsqueda de un Producto. [105]

Después en la imagen que se encuentra a cada extremo del campo correspondiente a producto y al número de documento contable se despliega una ventana auxiliar con un filtro de búsqueda secuencial mediante el nombre del producto el cual después de localizarlo procedemos a cargar la información que requiere la interfaz de registro del lote de medicación.

[105] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

De la misma forma que en la búsqueda y carga de un producto en la interfaz de registro de un lote de medicación procedemos a realizar la misma acción con el número de documento contable y realizamos la misma acción que con producto. Más adelante ahondaremos en el tema del DOCUMENTO CONTABLE.

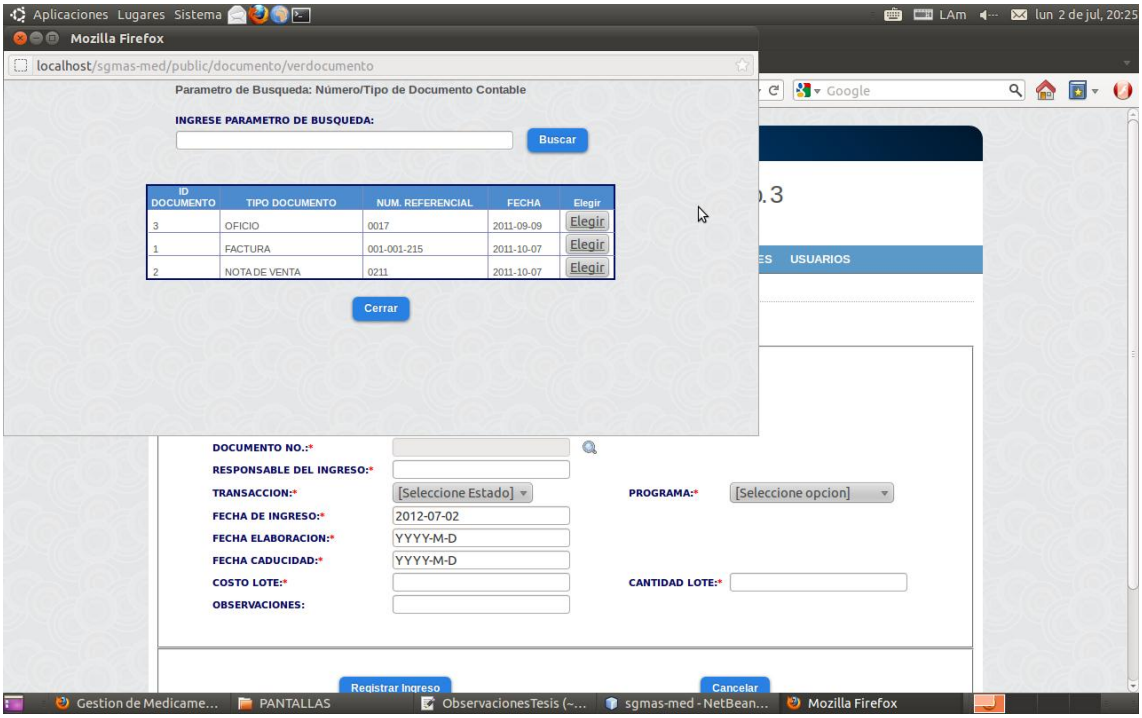
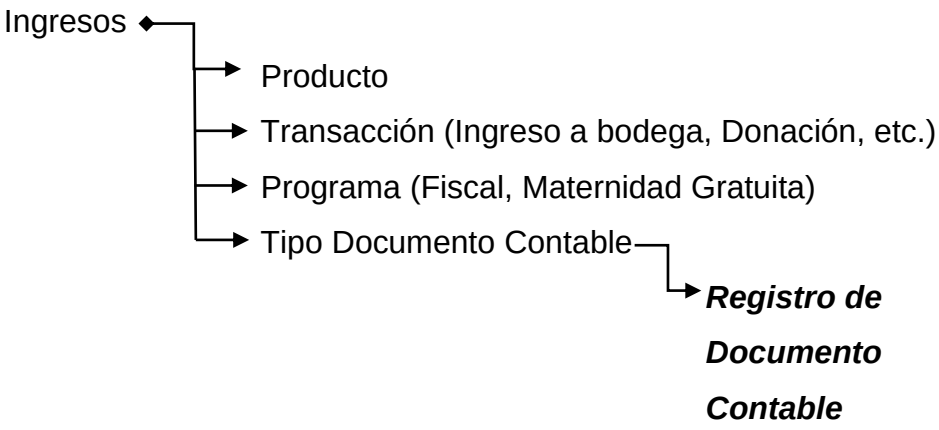


Figura 15.Parámetros de Búsqueda Documento Contable. [106]

Así mismo como en la pantalla de registro de un nuevo producto tenemos información obligatoria que debe estar antes almacenada en administración de catálogos, los cuales detallamos a continuación:

[106] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*



- **Documento Contable**

En ésta tabla de mantenimiento se guarda los tipos de documentos contables que se fueran a manejar en el ingreso de lotes de medicación para el efecto tenemos: FACTURA, SOLICITUD, etc.

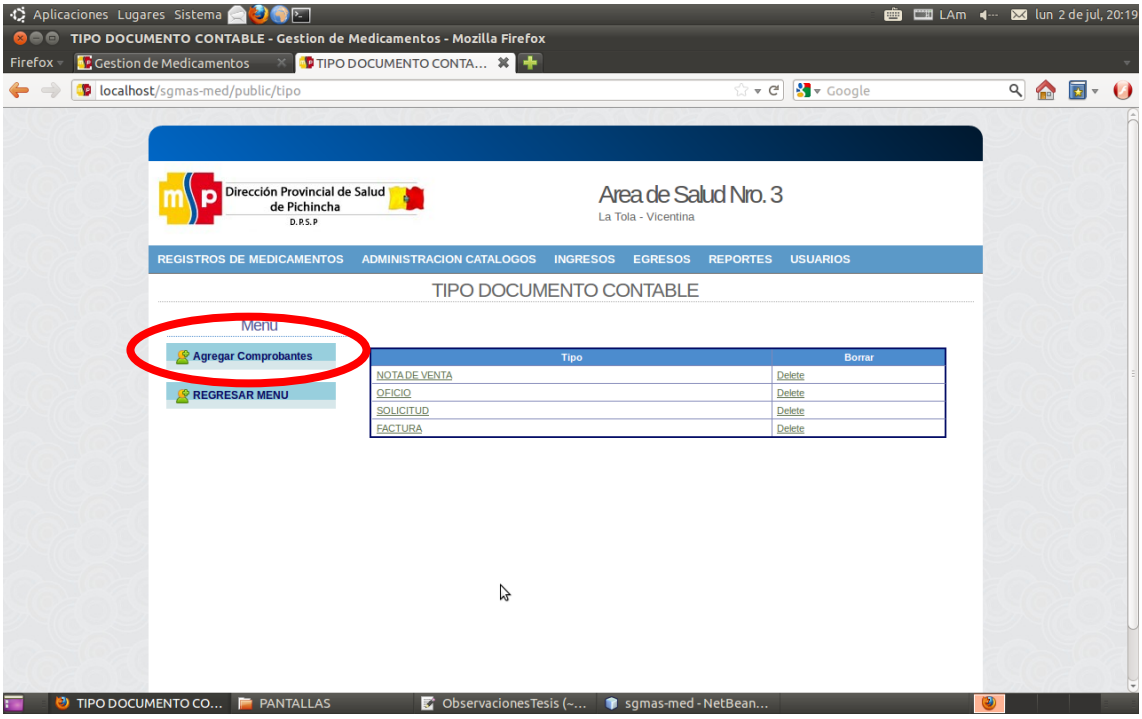


Figura 16. Tipo de Documento Contable [107]

[107] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Al no encontrarse registrado el tipo de documento contable requerido o necesitado se procede a registrar el mismo con el botón de la parte lateral izquierda “AGREGAR COMPROBANTES” después se despliega la siguiente pantalla de registro.

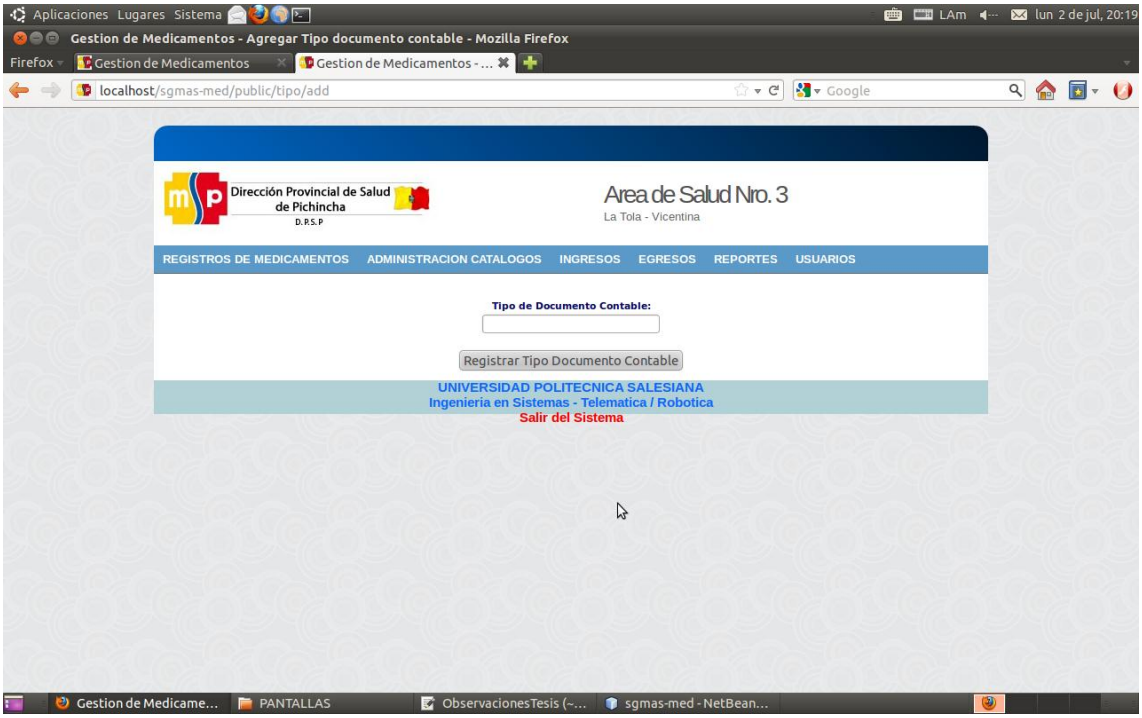


Figura 17.Agregar Comprobantes. [108]

- **Proveedores**

Se almacena información general de los proveedores que mediante algún tipo de documento contable abastecen de medicamentos a la bodega que va a ser registrada.

[108] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

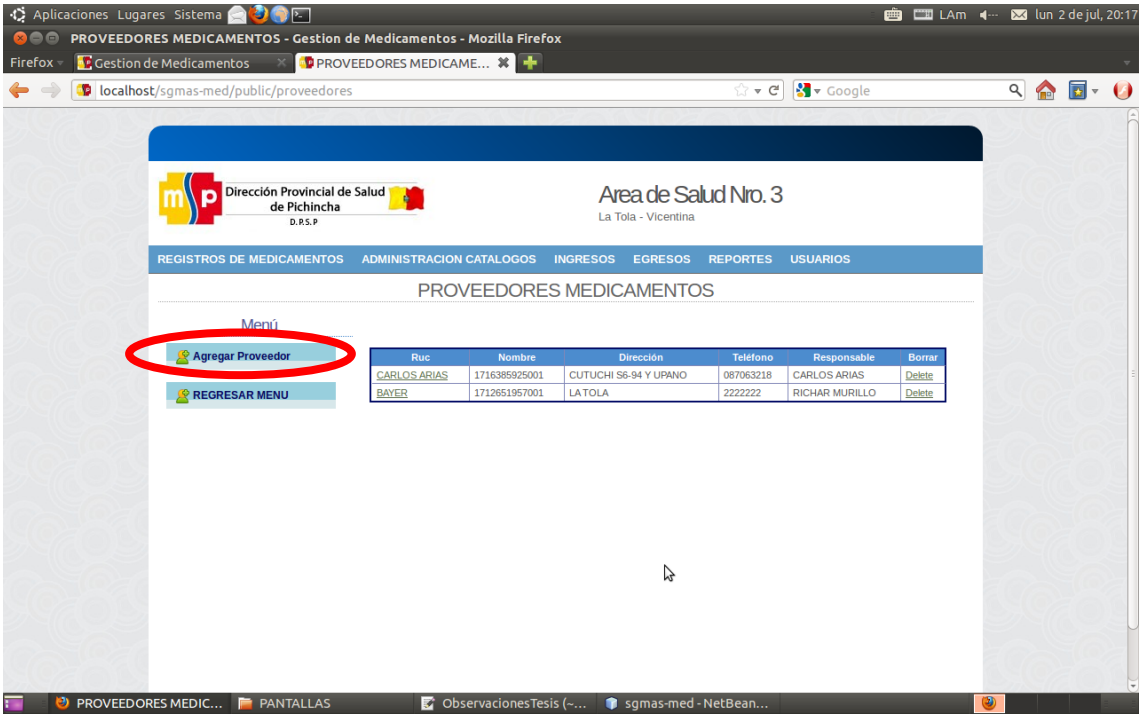


Figura 18. Proveedores. [109]

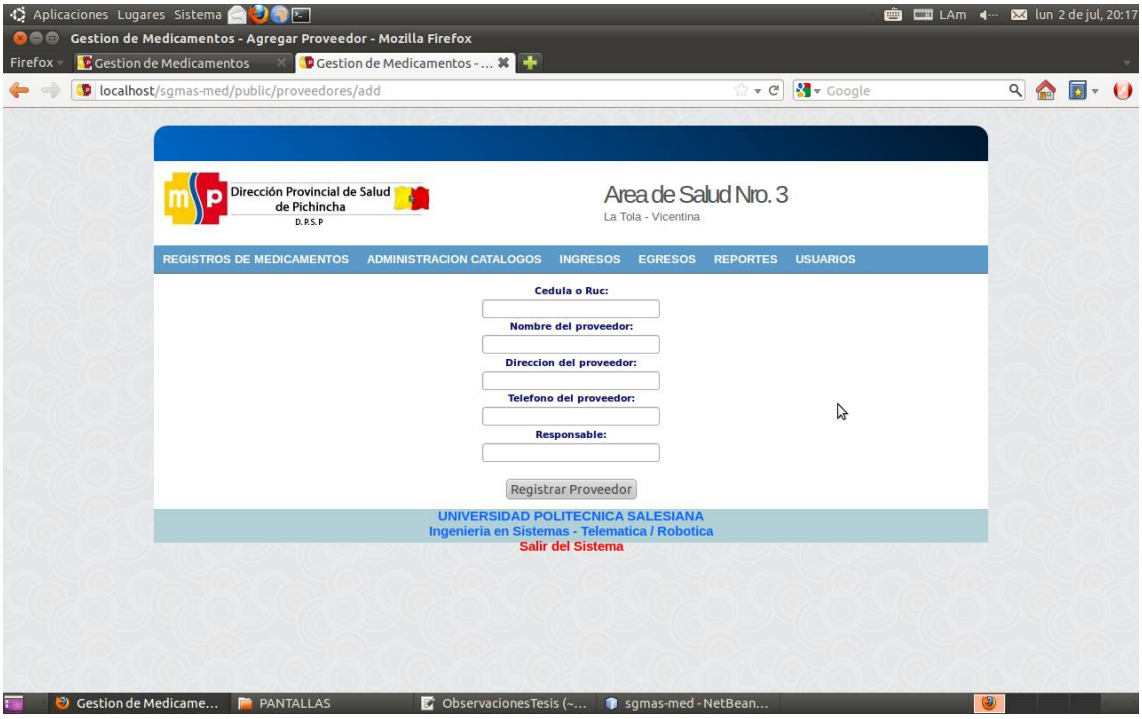


Figura 19.Ingreso Proveedor. [110]

[109] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*
[110] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

- **Registro de Documento Contable**

  Después de haber registrado el tipo de documento contable y los proveedores procedemos al REGISTRO DEL DOCUMENTO CONTABLE. Esta información es muy importante para la realización de reportes contables para un eficaz control de los lotes de medicación que ingresa y egresa de bodega con datos como Número de documento contable (Número de factura, nota de venta, número de referencia de alguna solicitud, memorando, etc.), el valor total de la medicación para luego ir segregando cada uno de los lotes a ingresar, su fecha de realización del documento, así como del proveedor y responsable del mismo; información suficiente para realizar futuras consultas que generen reportes para el control de los lotes de medicación.

  Nota: Es importante destacar que esta aplicación no realiza ningún tipo de contabilidad pero con la información que se almacena en el Sistema y los reportes que genera se puede llevar un eficiente control contable.

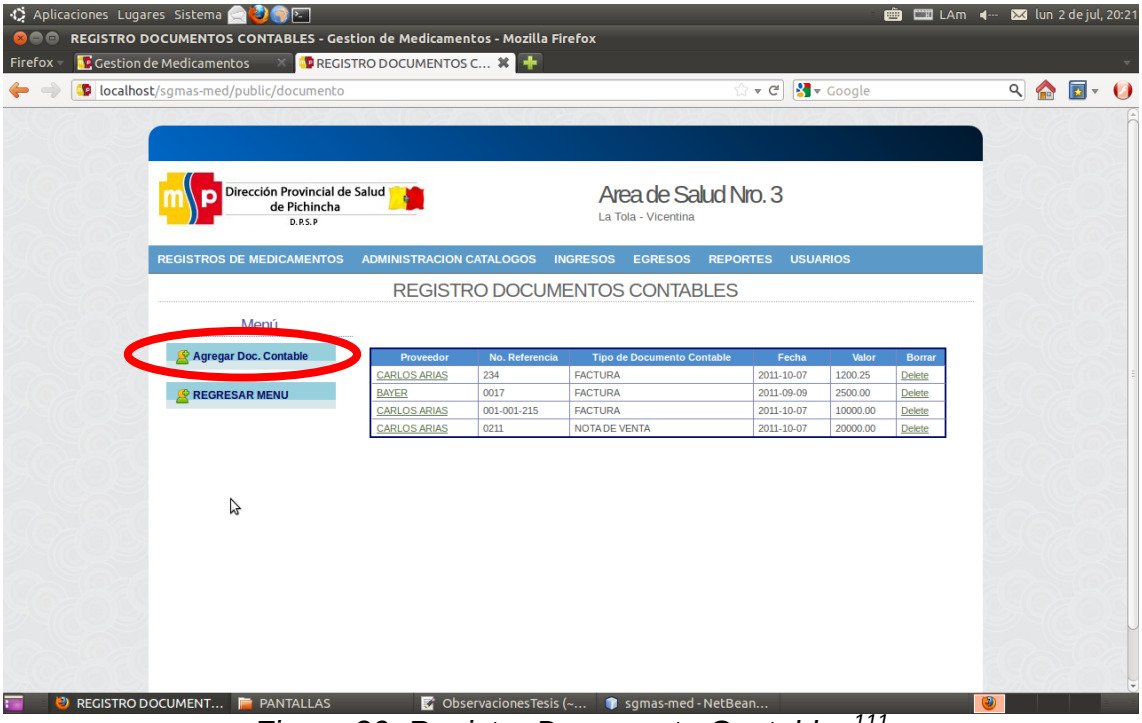


Figura 20. Registro Documento Contable. [111]

Figura 21. Ingreso Documento Contable. [112]

[111] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*
[112] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

6. ADMINISTRACIÓN DE CATÁLOGOS

Menú que nos permite administrar adecuadamente las tablas de mantenimiento del Sistema, que son tablas que por los tipos de datos y por la información que en ellos se guardan solo necesitan ingresarse una vez al Sistema pero que son de suma importancia para el funcionamiento de la Aplicación. Consta de:

- Bodega
- Familia
- Presentación
- Responsables
- Unidades (Referente a la unidad de medida que manejan los medicamentos).
- Proveedores
- Documento Contable
- Registro de Documento Contable

Figura 22. Administración de Catálogos. [113]

7. EGRESOS

Para realizar un egreso de lotes de medicación en el menú principal procedemos a escoger la opción de egresos el cual nos va a mostrar la siguiente pantalla:

[113] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Figura 23.Despacho de Medicación. [114]

En la pantalla principal de registro de egresos de lotes de medicación se presenta una tabla con el último egreso realizado por la bodega y en la parte lateral izquierda de la misma se encuentra un menú donde primero tenemos que realizar el encabezado de los despachos para luego proceder a realizar el registro de los medicamentos que se fueran a despachar para su posterior verificación física, autorización e impresión del comprobante con el detalle de los medicamentos entregados, el destinatario y el responsable de la realización del mismo.

El comprobante impreso se lo realiza individualmente dependiendo del programa al cual fue destinada la medicación indicando claramente el número de despacho, el número de productos despachados y el detalle de cada uno, con la firma del responsable de la realización del despacho.

[114] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

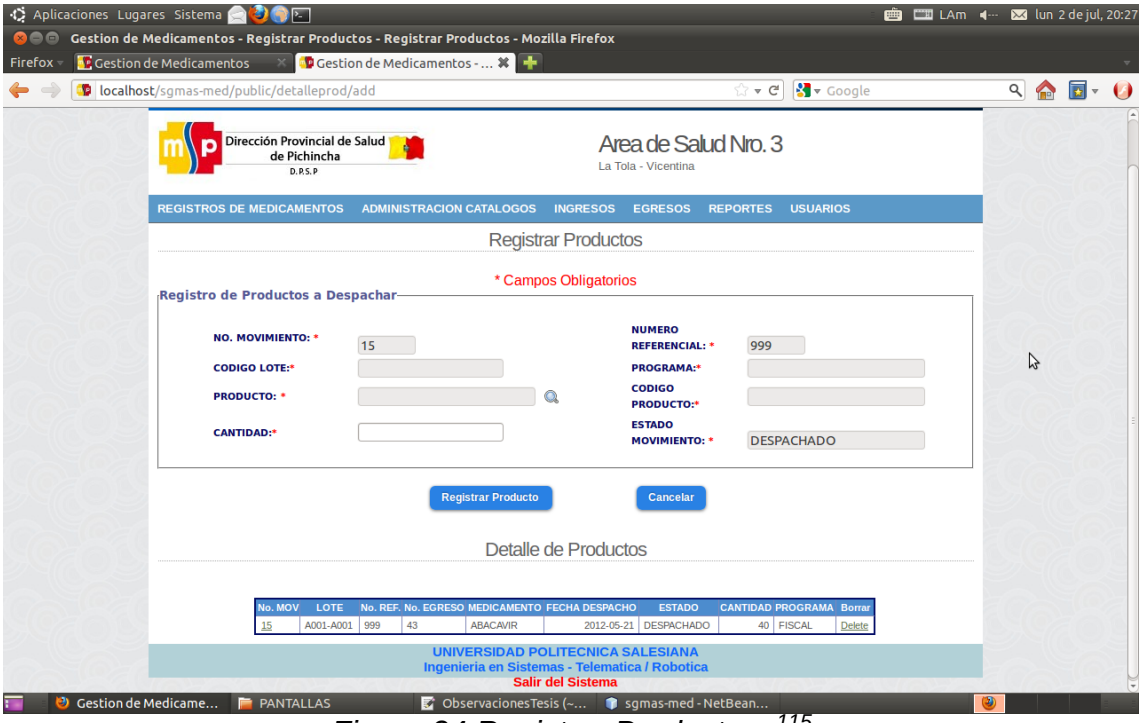


Figura 24.Registrar Productos. [115]

A la finalización de este proceso de registro del encabezado y de cada uno de los productos a despachar procedemos a la impresión del despacho; para hacerlo en el menú principal en la opción reportes seleccionamos la opción

[115] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Figura 25. Imprimir Despacho Generado. [116]

A continuación elegimos el programa a cual fueron destinados los medicamentos del último despacho realizado, inmediatamente se muestra el detalle de todos los medicamentos destinados a dicho programa para proceder a la impresión del comprobante.

[116] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

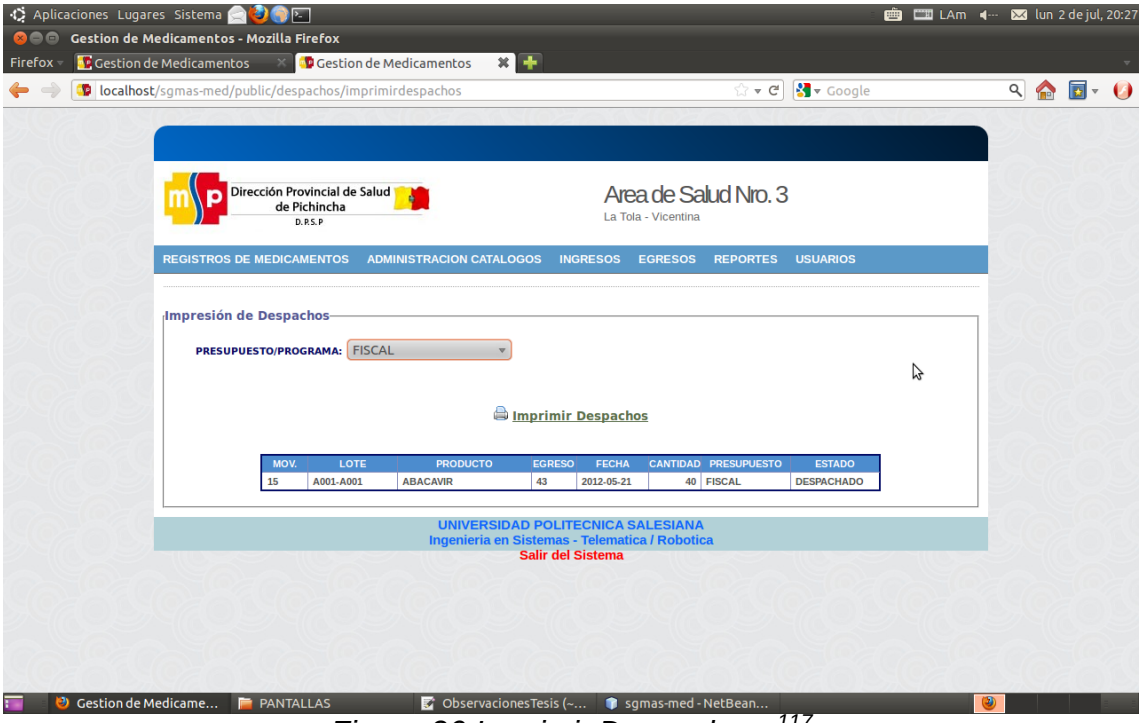


Figura 26.Imprimir Despachos. [117]

Impreso el comprobante y firmado por el responsable del despacho procedemos a cerrar este proceso.

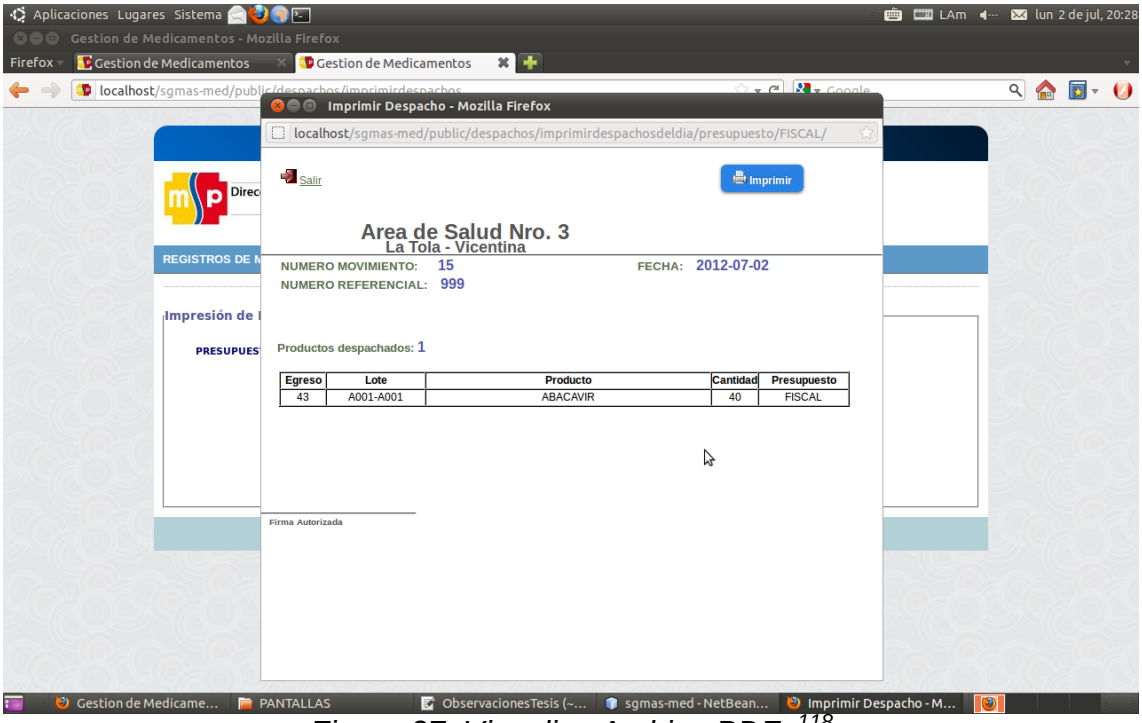


Figura 27. Visualiza Archivo PDF. [118]

[117] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

8. REPORTES

En el menú principal localizamos la opción reportes el cual despliega una lista de todos los reportes que genera la aplicación y son:

- Stock por lotes de medicación
- Reporte de despachos realizados por períodos de tiempo
- Despachos realizados
- Reporte de registro contable
- Control de ingresos realizados

Todos estos reportes dependiendo de la necesidad de los usuarios de la aplicación se pueden ir modificando, cambiando o eliminando si así lo requirieren con la respectiva autorización del administrador del Sistema ya que las herramientas utilizadas para el desarrollo de la aplicación es de libre distribución y el ADMINISTRADOR de la aplicación como de su implementación es el único responsable del mismo ya que el código fuente como de los MANUALES DE USUARIO y del PROGRAMADOR reposarán en sus manos.

[118] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*

Figura 28.Generar Reportes. [119]

[119] *Arias Carlos; Michelena Ma.José; SGMAS Módulo de Gestión Medicamentos; 2012.*