Universidad de las Ciencias Informáticas

Facultad 2

Título: *Plataforma de Gestión de Hardware y Software. Módulo: Mapa tecnológico.*

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

Autores: *Yulissa Torres Matos.*

Yanelis Morales González.

Tutores: *Ing. Denys Buedo Hidalgo.*

Ing. Vladimir Milián Núñez.

La Habana, Junio 2011, "Año 53 de la Revolución"

PENSAMIENTO

"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte."

DECLARACIÓN DE AUTORÍA

Declaramos que somos las únicas autoras de esta tesis y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2010.

Autor:

_____

Yanelis Morales González

Autor:

_____

Yulissa Torres Matos

Tutor:

_____

Ing. Denys Buedo Hidalgo

Co-Tutor:

_____

Ing. Vladimir Milián Núñez

AGRADECIMIENTOS

Agradecemos a nuestro Comandante en Jefe Fidel Castro Ruz por haber creado este maravilloso proyecto y a la Universidad de las Ciencias Informáticas por dejarnos ser parte de él.
A nuestros tutores Denys y Vladimir por la ayuda y la atención brindada.

A todos mis familiares, en especial a mis padres Idiana y Vladimir por darme siempre el mejor ejemplo y su apoyo incondicional en cada paso de mi vida.
A mi hermana Danelis por compartir sus 19 años de vida a mi lado y por dejarme ser su ejemplo a seguir.
A mi abuela Nena. Abu, dondequiera que estés sé que estás orgullosa de mí.
A mi padrasto Manuel por su paciencia y ayuda en todo momento.
A mis tíos, primos y mi abuelita Tomasa.
A Jorge, gracias por comprender, amar y respetarme tanto. Espero poder contar siempre contigo.
A la familia de Jorge que me quiere como a una más de su familia.
A todas mis amistades, a las que conozco desde mi primer año en la UCI y a los que después, me permitieron formar parte de sus vidas.
A Rayno, nunca voy a poder agradecerte lo suficiente por todo lo que hiciste por mí.
A mi compañera de tesis Yula por ser tan responsable, sin ella el resultado de este trabajo no hubiera sido posible y a su esposo Eliezer por ayudarnos siempre que lo necesitamos.
A todos los que aportaron sus experiencias y me ayudaron a realizar este sueño.

Gracias.

Yanelis.

Agradecimientos

Agradezco a todas aquellas personas que de una forma u otra contribuyeron al desarrollo de este trabajo, en especial:

A mi mami, por ser mi ejemplo en todo momento, por todo el apoyo brindado en estos cinco largos años, eres lo más lindo de mi vida.

A mi abuelita Loida, aunque no estés, sé que estarías muy orgullosa de mí.

A mis hermanitos Lisi y Yander, por ser mi alegría en todo momento de mi vida.

A mi esposo Eli, que ha sido mi apoyo desde el 1er año de mi carrera, gracias por siempre estar ahí, por regalarme los años más lindos de mi vida.

A toda mi familia, mis abuelitos Rigo, Wiliam y Aleida, mis tíos, Riguito, Weyler, Dayani, Roly, Yony, Daniel, Milene, gracias por su apoyo.

A mi papá, que sé que en el fondo me quiere.

A mis primitas, Leina y Weylena por ser la alegría de mi familia.

A la familia de Eli, muchas gracias por acogerme con tanto cariño.

A Odi (mi bobi), que ha sido mi hermanita, gracias por siempre estar en las buenas y las malas.

A Rayneri y Jorge, gracias por todo su apoyo, siempre les voy a estar eternamente agradecida.

A mi compañera de tesis, Yanelis (Moralito), por ser tan quisquillosa, sin ti no lo hubiésemos logrado.

A mis amistades, por haber sido como una familia, los voy a extrañar a todos.

A todas las personas que me han ayudado a hacer este sueño realidad. Muchas Gracias.

Yulissa.

DEDICATORIA

Dedico el resultado de este Trabajo de Diploma a las tres personas más importantes de mi vida: mi madre, mi padre y mi hermana. Los quiero más que a mí misma y no imagino un futuro sin ustedes a mi lado.

Yanelis.

Dedico mi tesis en especial a una persona que aunque ya no esté conmigo estaría muy orgullosa de mí, a ti abuelita Loida, te llevo en mi corazón para toda la vida. A mi mami, mis hermanitos Lisi y Yander y mi esposo Eli, que son el centro de mi vida, gracias por todo. Los quiero.

Yulissa.

RESUMEN

La investigación surge en el marco de trabajo de las aplicaciones desarrolladas por el Centro de Telemática de la Facultad 2. Este centro plantea la necesidad de poseer una herramienta que le ayude a centralizar la gestión de los recursos de sus laboratorios de producción. Para satisfacerla cuenta con un Sistema de Gestión de Recursos de Hardware y Software denominado GRHS. En esta herramienta el sistema servidor recibe los inventarios enviados por los clientes, almacena la información recibida en una base de datos y analiza si el inventario es una incidencia o no, de ser incidencia verifica si existe periodo de cambio para la incidencia, si no se ha definido periodo de cambio para ella, ejecuta alertas según la configuración establecida por el administrador de red, responsable del control sobre los recursos informáticos. La principal desventaja de la misma es que no se puede obtener una detallada visualización de las estaciones de los diferentes laboratorios de producción del centro. El desarrollo de la investigación está dirigido a solucionar dicha deficiencia y se propone la implementación de una interfaz web que muestre la información general, de hardware y software de las computadoras de los laboratorios de producción del centro de telemática de la facultad 2. Utilizando para ello las ventajas que ofrece el marco de trabajo Symfony, integrado a Doctrine y Ext JS. Se eligió RUP para guiar a través de sus fases y flujos el proceso de desarrollo. Los artefactos generados junto a las facilidades que brindó el framework escogido, hicieron posible obtener una aplicación web funcional y eficiente, que permite gestionar el inventario obtenido del sistema GRHS.

PALABRAS CLAVES

Inventario, Hardware, Software, aplicación web, interfaz web, laboratorios, Symfony, Doctrine, Ext JS y RUP.

ÍNDICE GENERAL

INTRODUCCIÓN

En la actualidad se ha alcanzado un gran auge en el uso de las Tecnologías de la Informática y las Comunicaciones como apoyo a los centros laborales, al auto aprendizaje y como medio de comunicación entre las personas.

Las actuales redes de telecomunicaciones se caracterizan por un constante incremento del número, complejidad y heterogeneidad de los recursos que las componen. Los principales problemas relacionados con la expansión de estas son la gestión de su correcto funcionamiento diario y la planificación estratégica de su crecimiento. La gestión de red, como conjunto de actividades dedicadas al control de los recursos, se ha convertido en un aspecto importante en el mundo de las telecomunicaciones.

Las aplicaciones informáticas que gestionan información son cada vez más utilizadas por las instituciones, debido a que permiten sustituir los procedimientos tradicionales de manipulación y control de la información.

A pesar de vivir en un mundo controlado por redes, los recursos que en ellas se encuentran no siempre son controlados e incluso desconocidos por sus respectivos propietarios o administradores de redes. A ello se une la problemática de que los inventarios se realizan de forma manual y no engloban información del software, porque son pocos frecuentes, la información se registra en papeles y la gestión de estos se encuentra completamente descentralizada.

A fin de dar solución a estas realidades, algunas empresas dedicadas a la producción de aplicaciones informáticas para la gestión de redes han dedicado un espacio al desarrollo de herramientas que proporcionen información de hardware y software informando al administrador de red sobre los cambios que son realizados en las computadoras que administra.

Actualmente se cuenta con una variada y novedosa gama de sistemas que realizan inventarios de hardware y software, algunos privados y otros con licencias de software libre. En la mayoría sólo se muestra un listado general de las estaciones de trabajo de la red.

La Universidad de las Ciencias Informática (UCI) como entidad dentro de Cuba que posee potencial y capacidad humana promueve el desarrollo de productos y servicios informáticos en aquellas ramas donde Cuba tiene un reconocido prestigio en el mundo. El desarrollo de Software y Servicios Informáticos se basa en la integración de los procesos de formación, investigación y producción en torno a una temática para convertirla en una rama productiva.

Dentro de las herramientas que se han producido en el Centro de Telemática de la Facultad 2 se encuentra el software denominado Sistema de Gestión de Recursos de Hardware y Software (GRHS). Este sistema posee una estructura cliente-servidor que automatizan los procesos de

obtención de información de hardware y software en redes de computadoras, y dispone de un sistema de gestión de alertas.

El mismo presenta dos divisiones lógicas totalmente distintas, pero dependientes una de la otra. La primera división lógica es una aplicación que procede a la obtención del inventario de los agentes o clientes del sistema, luego gestiona el proceso de alertas y guarda la información recibida de los agentes. La segunda división lógica es la aplicación de administración que gestiona la información obtenida de los clientes asociados al sistema y configura los procesos de alertas.

Su dificultad consiste en que, hasta el momento, no se puede obtener una detallada y clara visualización de un mapa de red de los diferentes laboratorios de producción del Centro de Telemática, puesto que desconoce a qué locales tecnológicos pertenecen las estaciones de trabajo de las cuales recolecta la información y actualmente, no tiene concebido una gestión de usuarios del sistema y sus permisos, lo que dificulta por ende, la gestión de los inventarios que recolecta.

Teniendo en cuenta las consideraciones analizadas anteriormente y la situación problémica se formula como **problema científico** a resolver: ¿Cómo construir un mapa de la red de los laboratorios de producción del Centro de Telemática, a partir de la información recogida en el Sistema de Gestión de Recursos de Hardware y Software GRHS?

Dado el problema científico se define como el objeto de estudio del presente trabajo: Mapas Tecnológicos de Redes, de donde se deriva el **campo de acción:** Mapa tecnológico del Sistema de Gestión de Recursos de Hardware y Software (GRHS) desarrollado en el Centro de Telemática de la Universidad de las Ciencias Informáticas.

A fin de resolver el problema anteriormente citado se planteó como **objetivo general:** Diseñar e implementar el Módulo: Mapa Tecnológico, del Sistema de Gestión de Hardware y Software desarrollado en el Centro de Telemática, usando tecnologías y herramientas libres y que permita visualizar toda la información recopilada por dicho sistema.

Ello da lugar a los siguientes **objetivos específicos**:

- ✓ Gestionar la obtención de información ofrecida por la el Sistema de Gestión de Recursos de Hardware y Software GRHS.
- ✓ Gestionar los usuarios, grupos, permisos, módulos y acciones en el sistema.
- ✓ Mostrar las computadoras distribuidas por locales en una aplicación web.

Las **tareas de investigación** para darle cumplimiento a los objetivos propuestos son:

- ✓ Estudio y análisis del estado del arte a nivel nacional e internacional de las herramientas que generan inventarios de Hardware y Software de las computadoras de la red.
- ✓ Análisis y diseño de la conexión a la Base de Datos del Sistema de Inventario de Hardware y Software para extraer información referente a las PCs de los laboratorios de producción del centro de telemática.
- ✓ Estudio y selección de las herramientas a utilizar para el desarrollo del sistema.
- ✓ Estudio y análisis de patrones de diseño y estilos arquitectónicos con el fin de elaborar un diseño robusto y flexible.
- ✓ Elaborar los diagramas de clases del diseño y diagramas de secuencia.
- ✓ Implementar los componentes diseñados y elaborar diagrama de componentes.
- ✓ Diseño de los casos de pruebas que serán aplicados sobre la aplicación.

Diseño Metodológico:

Durante la investigación se emplearon los siguientes métodos científicos:

Métodos teóricos:

- ✓ ***Analítico - Sintético:*** Se utilizó para el análisis de los documentos más importantes acerca de los Sistemas de Inventarios de Hardware y Software sintetizando sus características, ventajas y desventajas para el arribo a las conclusiones de la investigación.
- ✓ ***Histórico - Lógico****:* Necesario en el estudio de las tendencias actuales del desarrollo de mapas de redes y su evolución para, según sus principales ventajas, utilizarlo en el desarrollo de la aplicación.

Métodos empíricos:

- ✓ ***Observación:*** Escogido para realizar una valoración a partir de la percepción, en la etapa exploratoria de la realidad estudiada, a fin de determinar cómo ocurre el proceso de obtención de información de una base de datos y cuáles son los principales elementos que deben ser mostrados.

La investigación realizada se estructura en cuatro capítulos como sigue:

Capítulo 1.Fundamentación teórica.

Incluye un estudio del estado del arte de los Sistemas de Inventario de Hardware y Software a escala internacional y nacional, particularmente en la UCI, además, de las tendencias, tecnologías, metodologías y software usados en la actualidad para la elaboración de aplicaciones web en las que se apoya para la solución del problema que se presenta, sobre los que es necesario profundizar. Se presenta un estudio crítico y valorativo, no una mera reproducción de referencias y estadísticas.

Capítulo 2. Características del sistema.

Se realiza una argumentación del objeto de estudio, donde se analiza el problema y la situación problémica. Se lleva a cabo una descripción de los procesos que serán objeto de automatización. Por otra parte, se describe la propuesta del sistema: Mapas Tecnológicos de Redes y son mostradas, de forma general, su funcionalidades. Queda definido el Modelo de Dominio, los Casos de Uso del Sistema y la especificación de los Requerimientos del Software tanto funcionales como no funcionales.

Capítulo 3. Análisis y Diseño.

En este capítulo que se muestra la fase de análisis y diseño. Se elabora el Modelo Físico y Lógico de la Base de Datos con la descripción de algunas de sus tablas principales y se describen las clases que se desea persistan.

Capítulo 4. Implementación y Prueba.

El capítulo está dedicado a la descripción de los flujos de implementación y prueba, mostrando los diagramas necesarios para su comprensión, entre ellos el diagrama de despliegue de la aplicación y el diagrama de componentes. Así como el diseño de los casos de pruebas aplicadas al sistema Mapas Tecnológicos de Redes y los resultados que estos arrojaron.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

En este capítulo se plantea la fundamentación teórica del trabajo. En él se definen los conceptos más significativos asociados al dominio del problema. A partir de ahí se realiza el estudio del arte de los sistemas existentes que automatizan el proceso de inventario de hardware y software y construyan un de mapas de red. Por último, se especifican las tecnologías y herramientas seleccionadas para el desarrollo del software como son: Metodología de Desarrollo, Lenguaje de Modelado, Herramienta CASE, Lenguaje de Programación, Sistema Gestor de Base de Datos y Marco de Trabajo.

1.2. Conceptos asociados al dominio del problema.

1.2.1. Inventario.

Es la verificación periódica de las existencias de materiales, equipo, muebles e inmuebles con que cuenta una dependencia o entidad. (1)

1.2.2. Hardware.

Corresponde a las partes tangibles de un dispositivo electrónico: componentes electrónicos, electromecánicos y mecánicos; cables, gabinetes, periféricos y cualquier otro elemento físico (2)

1.2.3. Software.

Se conoce como software al soporte lógico de los dispositivos electrónicos; comprende el conjunto de componentes lógicos que permiten la realización de tareas específicas. (3)

1.2.7. Mapa Tecnológico de Red.

Representación gráfica de la topología de la red, incluyendo tanto conexiones internas como externas. Esta documentación puede apoyarse en un plano del edificio donde se instala la red. Suelen confeccionarse dos tipos de Mapas de Red: lógicos y físicos. (4)

1.2.8. Mapa de Red lógico.

En el mismo se indica la funcionalidad del objeto que describe, así como sus direcciones. (4)

1.2.9. Mapa de Red físico.

Es aquel donde se especifican las conexiones físicas (cableado) de una red. (4)

1.3. Estudio del arte.

Actualmente existen disímiles herramientas cuyo principal objetivo es realizar un control y manipulación de reportes de hardware y software en una red de computadoras. A continuación se exponen los distintos programas estudiados en el ámbito internacional.

Network Inventory Reporter

Es un programa que permite recopilar toda la información referente a aplicaciones y componentes de hardware de cada uno de los ordenadores que estén conectados bajo la misma Red de Área Local, incluso de aquellos que están conectados a través de una red inalámbrica. Esta herramienta permite obtener todos los datos de cada uno de los equipos sin necesidad de instalar nada en ellos, basta con instalar Network Inventory Reporter en el servidor. (5)

Net Support DNA

Es una completa solución modular que ofrece inventario de hardware, software y gestión de licencias. Presenta avisos detallados y personalizables, como la medición y el control de uso de aplicaciones y de Internet. Posee actualización automática por consulta de distribución de software a través de una LAN (en español Red de Área Local) o una WAN (en español Red de Área Amplia). Proporciona una puerta de enlace de comunicaciones integrada que le permite interactuar con sus activos con toda seguridad, incluso a través de Internet y en cualquier lugar, todo ello sin necesidad de una VPN (en español Red Virtual Privada), ni cambios en la red existente o en la configuración del cortafuego. Se integra con Active Directory y cuenta con Helpdesk basado en ITIL (en español Biblioteca de Infraestructuras de Tecnologías de Información). Esta herramienta corre bajo sistema operativo Windows. (6)

GLPI (en francés Gestionnaire Libre de Parc Informatique)

Distribuido bajo licencia GPL (en español Licencia Pública General), facilita la administración de recursos informáticos. Es una aplicación basada en tecnología Web y escrita en PHP (Preprocesador de hipertexto). Permite registrar y administrar los inventarios del hardware y el software de una empresa, optimizando el trabajo de los técnicos. No requiere de agentes, pues se trabaja vía Web con autenticación de usuarios. El software permite registrar la información de los inventarios, posibilita registrar solicitudes de servicio por parte de los usuarios y asignar la atención de dichas solicitudes al personal de soporte correspondiente. Las principales funcionalidades de GLPI están articuladas sobre dos ejes: El primero está relacionado con el inventario de todos los recursos informáticos, y el software existente, cuyas características se almacenan en bases de datos y el otro está basado en la administración e historiales de las diferentes labores de mantenimiento y los procedimientos llevados a cabo sobre esos recursos informáticos. (7)

OCS Inventory NG (Open Computer and Software Inventory Next Generation)

Es una herramienta que permite realizar inventario de los equipos de una red. Recolecta información diariamente del hardware y el software instalado en los ordenadores. Utiliza un agente

que ejecuta el inventario de los equipos clientes y un servidor de administración que centraliza los resultados del inventario. Las comunicaciones entre los agentes y el servidor de gestión se realiza mediante los protocolos HTTP / HTTPS. Es capaz de detectar todos los dispositivos activos de la red, tales como router, impresora de red y almacenar direcciones MAC e IP, soportando casi todas las plataformas disponibles en el mercado, tales como GNU/Linux, Windows, Mac os, Sun, IBM, AIX, entre otros, su licencia es GNU General Public License, versión (GNU GPLv2). (8)

Cacic

Desarrollado por el gobierno de Brasil, este sistema realiza el inventario de hardware y software por medio de tecnologías basadas en agentes inteligentes. Presenta un sistema disparador de alerta cuando se identifican cambios en componentes de hardware de cada computadora, así como la identificación de las carpetas e impresoras compartidas. Tiene soporte para las principales plataformas, como GNU/Linux y Windows y se encuentra disponible bajo la licencia GPL. (9)
A continuación se describen los distintos programas estudiados en el ámbito nacional.

Sistema de Gestión de Recursos de Hardware y Software GRHS
Aplicación cliente-servidor cuyo objetivo principal es centralizar en el servidor la información del hardware y el software instalados en las estaciones clientes. La aplicación cliente se encarga de obtener la información de hardware y software instalados y envía los datos recolectados servidor, determinando si hubo cambios en el hardware o el software a partir del inventario obtenido anteriormente. El sistema agente así como el sistema servidor son propiedad exclusiva de la Universidad de las Ciencias Informáticas (UCI).

Herramienta	MP	SWL	SA	MR	MR x LT
Network Inventory Reporter	no	no	no	no	no
Net Support DNA	no	no	si	si	no
GLPI	si	si	no	no	no
OCS Inventory NG	si	si	si	si	no
Cacic	si	si	si	no	no
GRHS	si	si	si	no	no

Tabla 1: Herramientas de Inventario de Hardware y Software.

En la tabla se muestran una comparación entre las principales herramientas estudiadas que realizan inventario de hardware y software, atendiendo a 5 criterios a tener en cuenta a la hora de

elegir un sistema colector de información: Si el sistema es multiplataforma, libre, cuenta con un sistema de alertas, posee un mapa de la red y si el mapa de la red está dividido por locales tecnológicos. El Network Inventory Reporter y el Net Support DNA son herramientas privativas y Cuba ha optado por el software libre como la única vía posible para su desarrollo informático. Network Inventory Reporter y GLPI no cuentan con sistema de alertas capaz de mantener informado a los administradores de cualquier cambio ocurrido en los ordenadores clientes, además de que el inventario se hace de forma periódica y no en el momento en que tiene lugar la incidencia. Net Support DNA y OCS Inventory NG en el mapa de red que presenta, las estaciones de trabajo no están distribuidas por locales sino que son mostradas en forma de listado, haciendo más compleja su gestión.

A partir de lo anteriormente planteado, se deriva la necesidad de realizar un módulo para el Sistema de Gestión de Recursos de Hardware y Software GRHS.

1.4. Metodología de Desarrollo de Software.

Dentro de las metodologías de desarrollo de software existen varias clasificaciones, como las ágiles y las robustas, dadas las condiciones del problema, el conocimiento del equipo de trabajo y el tiempo disponible se decidió utilizar como metodología el Proceso Unificado del Software (RUP por sus siglas en inglés)

1.4.1. Rational Unified Process (RUP).

RUP ofrece buenas prácticas para el desarrollo, la aplicación eficaz y la gestión de proyectos. Constituye una de las metodologías estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Está basado en UML como herramienta principal. Es adaptable a las necesidades del equipo de desarrollo y genera documentación de gran utilidad para la continuidad del proyecto. RUP al ser iterativo e incremental permitirá que tras cada iteración se pueda analizar la calidad del producto y que se vayan desarrollando prototipos de la aplicación con el fin de probarlos y detectar posibles fallos en su funcionamiento. (10)
En la figura se muestran los flujos de trabajo que propone RUP y las fases por la que se debe transitar en el desarrollo del software:

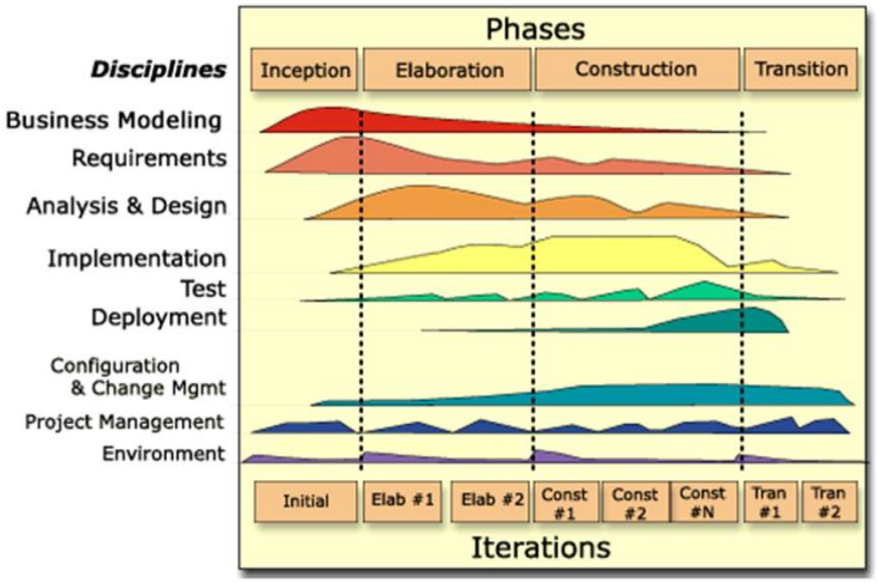

Figura 1: RUP en dos dimensiones. Organización mediante flujos de trabajo.

1.5. Unified Modeling Language (UML)

Se decide utilizar como Lenguaje de Modelado UML puesto que la Metodología de Desarrollo escogida se basa en él para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo del software. Es muy fácil de usar y cuenta con una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos. UML permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos. (11)

1.6. Herramienta Case.

Se puede definir a las Herramientas CASE como un conjunto de programas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Computer Aided Software Engineering (CASE) en su traducción al español significa Ingeniería de Software Asistida por Computación y se define también como un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. La estructura de las herramientas CASE se basa en la siguiente terminología:

- ✓ CASE de Alto Nivel: automatizan o apoyan las fases iníciales del ciclo de vida.
- ✓ CASE de Bajo Nivel: automatizan o apoyan las fases finales o inferiores del ciclo de vida.
- ✓ CASE Cruzado de Ciclo de Vida: apoyan las actividades que tienen lugar a lo largo de todo el ciclo de vida.

1.6.2. Visual Paradigm (Paradigma Visual).

Visual Paradigm For UML es una Herramienta Case Cruzado de Ciclo de Vida, fácil de usar, con soporte multiplataforma y que proporciona facilidades de interoperabilidad con otras aplicaciones. Permite la generación de código para Java y exportación como HTML, es fácil de instalar y actualizar. Soporta las últimas versiones de UML y la Notación y Modelado de Procesos de Negocios. Posee UML 2.1 Habilitado. Emplea una respuesta rápida con poca memoria utilizando moderadamente los tiempos del procesador, lo que le permite manejar grandes y complicadas estructuras de un proyecto en una forma muy eficiente y, que solo requiera de una configuración de escritorio. (13)

1.7. Lenguaje de Programación.

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos debe operar una computadora, cómo deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias.

Posteriormente se realiza un estudio de los distintos tipos de lenguaje de programación Web atendiendo a que los mismos se dividen, en lenguajes del lado del cliente y lenguajes del lado del servidor. (14)

Se decide utilizar JavaScript, XHTML y CSS como lenguajes de programación del lado del cliente, ya que JavaScript brinda la posibilidad de una mejor interacción con los usuarios a través del manejo de eventos, XHTML permite estructurar y etiquetar los documentos para su mejor organización y visualización, así como la utilización de objetos tales como imágenes dentro del texto, y CSS facilita crear variados estilos de páginas web, definiendo desde una plantilla CSS la visualización general de los sitios, haciendo las aplicaciones más atractivas y amigables para los usuarios. PHP como lenguaje de programación del lado del servidor, al ser un lenguaje desarrollado en el ámbito de los sistemas libres, bajo la licencia GNU. PHP es un lenguaje orientado a objetos, simple, elegante y con seguridad en el tratamiento de tipos, y cuenta con las principales funciones para lograr una solución factible al problema en cuestión.

1.7.1. Lenguaje del lado del Cliente.

Los lenguajes de lado cliente son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pre-tratamiento. Dentro de este lenguaje se encuentran: HTML, Java y el JavaScript. (15)

1.7.1.1. HTML.

Hyper Text Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web e indica al navegador dónde colocar cada texto, imagen o video y la forma que tendrán estos al ser colocados en la página. (15)

1.7.1.2. JavaScript.

JavaScript es un lenguaje de programación interpretado, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario.

Al igual que Java, JavaScript es un lenguaje orientado a objetos puesto que dispone de herencia. Esta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases bases (prototipos) y extendiendo su funcionalidad. Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. (15)

1.7.1.3. CSS.

CSS son las siglas de Cascading Style Sheets, en español Hojas de Estilo en Cascada. Permite crear páginas web. Gracias a esta los desarrolladores se involucran en los resultados finales de la página, pudiendo incluir márgenes, tipos de letra, fondos, colores e incluso pueden definir los estilos en un archivo externo. Si en algún momento se quiere cambiar alguno de los estilos, automáticamente se actualizan todas las páginas vinculadas al sitio.

Este estilo propone varias ventajas dentro de las cuales cabe destacar:

- ✓ Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- ✓ El documento HTML en sí mismo es más claro de entender y se consigue reducir su tamaño (siempre y cuando no se utilice estilo en línea). (14)

1.7.2. Lenguaje del lado del Servidor.

Los lenguajes de lado servidor que son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Ejemplos de estos son: PERL, ASP, PHP, entre otros. (15)

1.7.2.2. PHP.

Sus siglas vienen de Hypertext Pre-processor (Pre-Procesador de Hipertextos) es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas Web dinámicas. Es usado principalmente en interpretación del lado del servidor 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. Generalmente se ejecuta en un servidor web, tomando el código PHP como entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Es un lenguaje de alta potencia, fácil de usar e incluye la programación orientada a objetos. (16)

Principales usos del PHP son:

- ✓ Programación de páginas Web dinámicas, habitualmente en combinación con motores de bases de datos.
- ✓ Programación en consola, al estilo de Perl o Shell scripting.
- ✓ Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y GTK lo que permite desarrollar aplicaciones de escritorio en los sistemas operativos en los que está soportado.

Ventajas de PHP:

- ✓ Es un lenguaje multiplataforma, libre.
- ✓ Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- ✓ Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- ✓ Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- ✓ Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Permite las técnicas de Programación Orientada a Objetos.
- ✓ Biblioteca nativa de funciones sumamente amplia e incluida.

1.8. IDE de Desarrollo.

Un Entorno de Desarrollo Integrado (IDE), es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Provee un marco de trabajo amigable para los lenguajes de programación. (14)

1.8.2. Netbeans.

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Es un producto libre y gratuito, sin restricciones de uso. Provee de una estructura para los proyectos y propone un esqueleto para organizar el código fuente. El editor de este IDE integra lenguajes como HTML, JavaScript y CSS.

Entre sus principales características se encuentran:

- ✓ Integración con Symfony y ZenFramework.

Esta es una de las características por las que se selecciona NetBeans, además porque es posible dejar de lado la consola de comandos de Symfony y centrarse en desarrollar en el IDE.

- ✓ Editor de Código Fuente

Considerable mejora en su editor, sobre todo en el editor de PHP, es mucho más ágil y a la vez robusto, contiene más ayuda en línea, reconocimiento de sintaxis y todo lo que provee la última versión de PHP.

- ✓ Depuración de PHP

La salida del programa PHP aparece en una pantalla de línea de comandos en el IDE y se puede inspeccionar el código HTML generado sin tener que cambiar a un navegador.

Por lo anteriormente analizado se escoge como herramienta NetBeans como editor para PHP, ya que se sustenta fundamentalmente en su soporte, posibilidades de depuración y pruebas de PHP.

1.9. Sistema Gestor de Base de Datos.

Los Sistemas de Gestión de Bases de Datos (SGBD) pueden definirse como un paquete generalizado de software, que se ejecuta en un sistema computacional anfitrión, centralizando los accesos a los datos y actuando de interfaz entre los datos físicos y el usuario. Se componen de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de

consulta y tienen como propósito general manejar de manera clara, sencilla y ordenada un conjunto de información.

La Plataforma de Gestión de Hardware y Software. Módulo: Mapa tecnológico, se conectará a la base de datos del Sistema de Inventario de Hardware y Software, por tanto el sistema gestor de Base de Datos que se utiliza es en el que ese sistema almacena sus datos: PostgreSQL que es compatible con almacenamiento de objetos binarios de gran tamaño, incluyendo imagines, música y videos. Presenta interfaces de programación nativa para C / C + +, Java, .NET, Perl, Python, Ruby, Tcl, ODBC, entre otros, además posee buena documentación. Cuenta con sofisticadas funciones como, respaldo incremental, espacio de tablas, replicación asincrónica, transacciones anidadas (puntos de retorno), backups en caliente y un planificador de consultas sofisticadas / optimizador. Es compatible con conjuntos de caracteres internacionales, codificación de caracteres multibyte, Unicode, y es consciente de la configuración regional para la clasificación y el formato. Además de que PostgreSQL es un gestor objeto-relacional de código abierto sumamente potente e ideal para grandes sistemas. Es altamente escalable, tanto en la enorme cantidad de datos que puede manejar como en el número de usuarios concurrentes que puede acomodar. (17)

1.10. Framework.

Un framework es una estructura conceptual y tecnológica de soporte definida, con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Incluye soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Ventajas de utilizar un framework:

- ✓ El programador no necesita plantearse una estructura global de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar".
- ✓ Facilita la colaboración. Existen herramientas (utilidades, librerías) adaptadas al framework concreto para facilitar el desarrollo.

En la actualidad existen una gran variedad de framework entre los más usados se encuentran Symfony, Mojavi, CakePHP, Zend Framework, Prado, CodeIgniter, PHPOnTrax, Phrame y Seagull.

1.10.1. Symfony.

Se decide utilizar como framework Symfony. Este proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además,

automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas Unix, Linux, entre otras, como en plataformas Windows. A continuación se muestran algunas de sus características:

- ✓ Fácil de instalar y configurar en la mayoría de las plataformas.
- ✓ Independiente de un sistema gestor de bases datos.
- ✓ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ✓ Basado en la premisa de convenir en vez de configurar, en la que el desarrollador solo debe configurar aquello que no es convencional.
- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la Web.
- ✓ Preparado para aplicaciones empresariales y adaptables a las políticas de arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- ✓ Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- ✓ Fácil de extender, permite la integración con librerías desarrolladas por terceros. (18)

1.10.2. Ext JS.

Ext JS es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM.. Desde la versión 1.1 puede ejecutarse como una aplicación independiente.

Dispone de un conjunto de componentes para incluir dentro de una aplicación web, como: cuadros y áreas de texto, campos para fechas, campos numéricos, comboboxs, radiobuttons, checkboxs, editor HTM, elementos de datos, árbol de datos, pestañas, barra de herramientas, menús al estilo de Windows, paneles divisibles en secciones, sliders entre otros. (19)

1.10.2. Doctrine.

Doctrine es un Mapeador de Objeto-Relacional (ORM) para PHP que proporciona la persistencia a los objetos. Una de sus características clave es la opción de escribir las consultas de base de datos en un dialecto orientado a objetos de propiedad SQL llamada Doctrine Query Language (DQL). Genera clases de objetos de una base de datos existente, y el programador puede

especificar las relaciones y agregar funcionalidades personalizadas a las clases generadas. No hay necesidad de generar o mantener complejos esquemas de bases de datos XML, como se ve en muchos otros marcos.

Entre sus principales características se puede mencionar:

- ✓ Soporta la aplicación de varios comportamientos a las modelos.
- ✓ Tiene un motor de búsqueda fulltext.
- ✓ Usando las librerías permite escribir un simple archivo *YAML*, o bien, código *PHP* si se prefiere. Si se usa *YAML*, tiene algunos métodos que se pueden llamar en el código, o se descarga una interface de línea de comandos para construir las modelos. (20)

1.11. Conclusiones parciales.

A partir del análisis realizado durante el desarrollo del capítulo fueron mencionadas algunas deficiencias en los sistemas estudiados como Network Inventory Reporter, Net Support DNA, GLPI, OCS Inventory NG, Cacic y el Sistema de Gestión de Recursos de Hardware y Software GRHS, que reflejan carencia de funcionalidades, problemas de soporte e incompatibles con la filosofía código abierto. Ante la ausencia de aplicaciones que cumplan con las características básicas requeridas para resolver el problema surge la necesidad de desarrollar una aplicación que permita trazar un mapa de la red de los laboratorios de producción del centro de telemática y permita conocer información sobre las estaciones de trabajo almacenadas en la base de datos del sistema GRHS. Se escogieron las herramientas y metodologías para el desarrollo de la aplicación, UML como Lenguaje de Modelado, RUP como Metodología de Desarrollo, Visual Paradigm como Herramienta CASE, PHP como Lenguaje de Programación soportado por un IDE de Desarrollo como NetBeans, las cuales deben transformarse en calidad, reducción de tiempo y costos al finalizar el proceso de desarrollo.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción.

En el presente capítulo se plantean los objetivos estratégicos del Centro de Telemática de la facultad 2, se especifican los procesos que serán objetos de automatización, así como la propuesta de sistema para la implementación de un mapa de red para los laboratorios de producción de dicho centro, haciendo énfasis en el flujo de procesos del mismo y su arquitectura. Se plasman los requerimientos funcionales y no funcionales que deben cumplir el software. Por último se muestra el Diagrama de Casos de Uso del Sistema y se describen algunos de los casos de uso más críticos.

2.2. Objeto de automatización.

- ✓ Proceso de creación de Mapas de Redes de Computadoras.

2.3. Propuesta de sistema.

La plataforma que se propone desarrollar automatizará la gestión de los recursos de la red existentes en los locales tecnológicos del Centro de Telemática de la facultad 2. La misma permitirá la visualización de la información del sistema GRHS de forma dinámica y será capaz de gestionar tanto los locales tecnológicos como las estaciones de trabajo.

La distribución de las computadoras por locales garantiza una gestión centralizada de sus recursos, posibilitando que la búsqueda sea más rápida y efectiva. Por otra parte, la aplicación propuesta resolverá la problemática de la gestión de los usuarios del sistema, los grupos de usuarios y los permisos de cada uno, que en estos momentos el sistema GRHS no tiene concebido. De esta forma se les brinda a los administradores una herramienta segura, a la cual sólo podrán acceder los usuarios por ellos definidos y ejecutarán las acciones que le sean permitidas.

En la aplicación propuesta son definidas políticas de seguridad, otorgando a cada usuario los derechos que le corresponden. Existirán 3 tipos de usuarios: usuario, administrador y editor. Cada usuario que accede al sistema deberá autenticarse antes de realizar alguna acción.

2.4. Modelo de Dominio.

Debido que el negocio actual presenta un bajo nivel de estructuración, sin poder identificarse las personas que desarrollan las distintas actividades en cada uno de los procesos, se decide realizar el modelo de dominio que RUP propone para estos casos. En este modelo se relacionan los principales conceptos identificados que se manejarán durante el desarrollo del sistema,

contribuyendo a que los usuarios, implementadores e interesados en la aplicación, puedan manejar un vocabulario común, con el objetivo de lograr una mejor comprensión del problema, permitiendo la correcta captura de requisitos y el desarrollo exitoso del sistema. (11)

2.4.1. Conceptos del Modelo de Dominio.

Estación de Trabajo: Es la unidad completa y tiene una ubicación física en el mapa tecnológico de red. Facilita a los usuarios el acceso a los servidores y periféricos de la misma.

Inventario: Proceso de identificación y categorización de los recursos de información de una forma sistemática.

Inventario General: Consiste en la recolección de información relacionada con la estación como: Ubicación Física, Dirección IP, Dirección de Puerta de Enlace, Dirección Mac, Nombre de la PC Agente y Usuarios de PC Agente.

Inventario de Hardware: Consiste en la recolección de la información de hardware de la PC que incluye información de: Motherboard, Memoria, Disco Duro, Monitor, Teclado, Ratón, Procesador y BIOS.

Inventario de Software: Consiste en la recolección de información asociada a los paquetes instalados en la PC, esta incluye: Nombre del paquete e información referente al Sistema Operativo en el que corre la aplicación: Número de Serie, Versión y Nombre de Producto.

Local tecnológico: Todo lugar que cuente con al menos una estación, dedicada al diseño, desarrollo o gerencia de los sistemas informáticos computarizados, pueden ser oficinas o laboratorios de producción.

Mapa Tecnológico de Red: Representación gráfica de la topología de la red.

SMTR: Sistema de Mapa Tecnológico de Red.

Usuario: Actor que interactúa con el sistema. Según los permisos que tenga asignado podrá realizar determinadas acciones sobre el sistema.

A continuación se muestra el Modelo de Dominio del sistema.

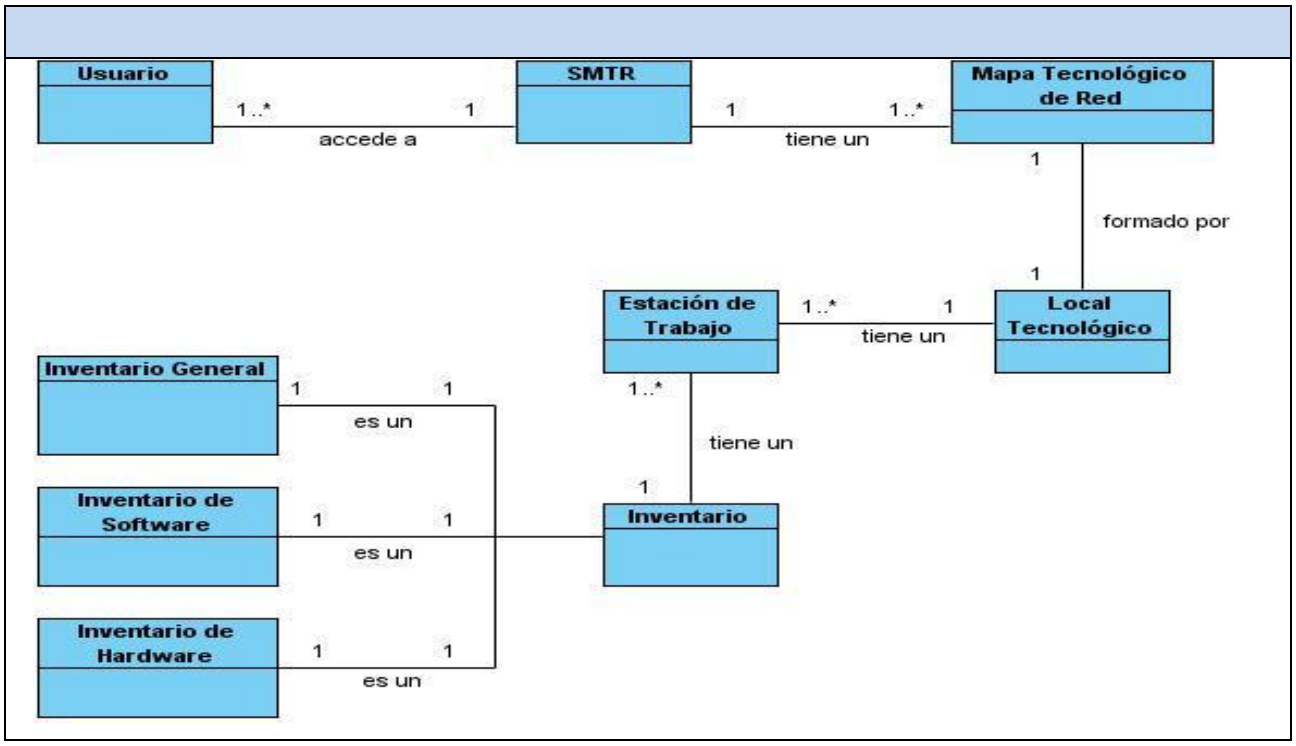

Figura 2: Modelo de Dominio.

2.4.2. Descripción del Modelo de Dominio.

Los usuarios podrán acceder al Sistema Mapas Tecnológicos de Redes y realizar funcionalidades en correspondencia de los permisos asignados. Este sistema está constituido por uno o varios mapas tecnológicos de redes que está formado por un local tecnológico. Un local se constituye por una o varias estaciones de trabajo las cuales tienen un inventario general, uno de hardware y otro de software.

2.5. Especificación de los requisitos de software.

Los requisitos son las condiciones o capacidades que deben ser alcanzadas o poseídas para que un sistema satisfaga las necesidades por las cuales fue creado. En otras palabras, definen lo que el sistema debe hacer. El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto.

2.5.1. Requerimientos Funcionales.

Los requisitos funcionales no son más que condiciones o capacidades funcionales que el sistema debe de tener. Definen actividades internas del software, como manipulación de datos, flujos de información en dependencias de situaciones predefinidas por eventos lógicos. A continuación se muestran los requisitos funcionales para el sistema propuesto:

RF1. Autenticar Usuario.

RF2. Gestionar Usuario.

- RF2.1. Adicionar Usuario.
- RF2.2. Editar Usuario.
- RF2.3. Eliminar Usuario.
- RF2.4. Mostrar Usuario.

RF3. Listar Usuarios.

RF4. Gestionar Grupo.

- RF4.1. Adicionar Grupo.
- RF4.2. Editar Grupo.
- RF4.3. Eliminar Grupo.
- RF4.4. Mostrar Grupo.

RF5. Listar Grupos.

RF6. Asignar Permiso a Grupo.

RF7. Asignar Usuario a Grupo.

RF8. Asignar Permiso a Usuario.

RF9. Gestionar Acción.

- RF9.1. Adicionar Acción.
- RF9.2. Editar Acción.
- RF9.3. Eliminar Acción.
- RF9.4. Mostrar Acción.

RF10. Listar Acciones.

RF11. Gestionar Módulo.

- RF11.1. Adicionar Módulo.
- RF11.2. Editar Módulo.
- RF11.3. Eliminar Módulo.
- RF11.4. Mostrar Módulo.

RF12. Listar Módulos.

RF13. Gestionar Local Tecnológico.

- RF13.1.Adicionar Local Tecnológico.
- RF13.2.Editar Local Tecnológico.
- RF13.3. Eliminar Local Tecnológico.
- RF13.4 Mostrar Local Tecnológico.

RF14. Listar Locales Tecnológicos.

RF15. Asignar Estaciones de Trabajo a Local Tecnológico.

RF16. Gestionar Tipo de Local.

RF16.1. Adicionar Tipo de Local.

RF16.2. Editar Tipo de Local.

RF16.3. Eliminar Tipo de Local.

RF16.4. Mostrar Tipo de Local.

RF17. Listar Tipos de Locales

RF18. Consultar Estación de Trabajo.

2.5.2. Requerimientos No Funcionales.

A continuación se reflejan las cualidades o propiedades que el producto debe tener para lograr características que lo hagan rápido, confiable, atractivo, usable y seguro. Evidencian las restricciones del entorno, de la implementación, el rendimiento del sistema, así como las dependencias de plataformas entre otros aspectos.

2.5.2.1. Requerimientos de software.

- ✓ **Requisitos óptimos del software del servidor.**

Sistema operativo Windows o Linux.

Para la implantación del sistema se requiere un servidor WEB Apache versión 2.0 o superior.

Servidor de base de datos PostgreSQL 8.2.x

- ✓ **Requisitos óptimos del software del cliente**

Navegador WEB con capacidad de interpretación de JavaScript (Mozilla Firefox, Opera, Google Chrome o Safari).

2.5.2.2. Requerimientos de hardware.

Si se tratase de una pequeña red, con servicio para algunas decenas de estaciones de trabajo se puede emplear:

- ✓ Microprocesador Pentium IV.
- ✓ 512 MB de memoria RAM.
- ✓ 1 GB de espacio libre en el disco duro.

Incluso pudiera operar correctamente en condiciones inferiores; sin embargo, si el tamaño de la red aumentara y se tratase ya de algunos miles de estaciones de trabajo, el cambio más significativo estaría en aumentar la capacidad de almacenamiento, además debe emplearse un procesador con capacidad de multiprocesamiento. Por otra parte se hace necesario aumentar la capacidad de memoria RAM hasta un rango de 1 a 4 GB.

2.5.2.3. Restricciones en el diseño y la implementación.

- ✓ La aplicación se desarrollará utilizando el lenguaje de programación PHP, como IDE de desarrollo Netbeans, Servidor Web Apache y como SGBD PostgreSQL.
- ✓ Como arquitectura general del sistema se define la utilización de la arquitectura MVC ya que separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

2.5.2.4. Requerimientos de apariencia o interfaz externa.

- ✓ El sistema debe tener una apariencia amigable propia de una interfaz web con colores de bajos tonos.
- ✓ La interfaz debe permitir el fácil manejo para todo tipo de usuarios
- ✓ Las páginas deben mostrar solo la información e imágenes necesarias.
- ✓ Diseño sencillo y claro, con reconocimiento visual a través de elementos visibles que identifiquen cada una de sus funcionalidades.

2.5.2.5. Requerimientos de usabilidad.

- ✓ El sistema debe ser de fácil manejo o manipulación aún para personas que no tengan muchos conocimientos de trabajo en computadoras y ambientes WEB.

2.5.2.6. Requerimientos de rendimiento.

- ✓ Rápido acceso a la información con tiempos de respuesta relativamente cortos de, aproximadamente 2 segundos.
- ✓ El sistema debe requerir un consumo mínimo de recursos del servidor y las PCs clientes.

2.5.2.7. Requerimientos de seguridad.

- ✓ **Confidencialidad**

- Garantizar que la información sea editada o modificada por las personas que tienen permisos para hacerlo.
- Se debe tener un mecanismo de seguridad basado en el modelo (Autentificación, Autorización y Auditoría).
- Mostrar mensajes de error en caso que el usuario intente autenticarse con datos incorrectos.

- ✓ **Integridad**

- Prevenir posibles fallos y recuperarse ante la afectación de alguno.

- Para lograr la recuperación ante fallos deberá existir un mecanismo de recuperación de los datos, como salva automáticas de la base de datos o copias de respaldo que puedan contribuir posteriormente al restablecimiento ante pérdida total de información.
- Debe existir un personal capacitado para realizar las salvas de la base de datos. Las mismas se deben almacenar en discos duros externos que deben ser guardados en un local con una adecuada protección y con acceso sólo al personal autorizado. Primeramente debe hacerse una salva total de la base de datos y los fines de semana una salva incremental.
- Protección contra acciones no autorizadas que puedan afectar la integridad de los datos.
  - ✓ **Disponibilidad**
- Se le garantizará el acceso a los usuarios autorizados a la información en un tiempo relativamente corto, o sea, la seguridad implantada en el sistema no afectará el tiempo de respuesta a las solicitudes de los usuarios.

2.5.2.8. Requerimientos de Portabilidad.

- ✓ El sistema es independiente de plataforma, podrá ser usado bajo los sistemas operativos Windows y Linux.

2.5.2.10. Requerimientos legales:

- ✓ Propiedad exclusiva de la Universidad de las Ciencias Informáticas (UCI).

2.6. Definición de los actores.

Actores	Justificación
Usuario	Persona autenticada en el sistema que interactúa con él, para consultar información de locales tecnológicos y estaciones de trabajo.
Administrador	Persona autenticada en el sistema que interactúa con él. Gestiona locales tecnológicos, tipos de locales, estaciones de trabajo, grupos, usuarios, permisos, acciones y módulos.
Editor	Persona autenticada en el sistema que interactúa con él. Modifica información de locales tecnológicos.

Tabla 2: Actores del Sistema.

2.7. Listado de casos de uso.

CU-1	Autenticar Usuario
Actor	Usuario, Administrador, Editor.

Descripción	Es iniciado cuando un usuario del sistema, dígase Administrador, Usuario o Editor acceda al sistema en correspondencia con los permisos definidos.
Referencia	RF1

Tabla 3: Listado de casos de uso: Autenticar Usuario.

CU-2	Gestionar Usuario.
Actor	Administrador.
Descripción	Permite al administrador Adicionar, Editar, Eliminar o Mostrar los usuarios del sistema.
Referencia	RF2, RF2.1, RF2.2, RF2.3, RF2.4.

Tabla 4: Listado de casos de uso: Gestionar Usuario.

CU-3	Listar Usuarios.
Actor	Administrador.
Descripción	Permite al administrador obtener un listado de los usuarios del sistema.
Referencia	RF3.

Tabla 5: Listado de casos de uso: Listar Usuarios.

CU-4	Gestionar Grupo.
Actor	Administrador.
Descripción	Permite al administrador Adicionar, Editar, Eliminar o Mostrar los grupos del sistema.
Referencia	RF4, RF4.1, RF4.2, RF4.3, RF4.4.

Tabla 6: Listado de casos de uso: Gestionar Grupo.

CU-5	Listar Grupos.
Actor	Administrador.
Descripción	Permite al administrador obtener un listado de los grupos del sistema.
Referencia	RF5.

Tabla 7: Listado de casos de uso: Listar Grupos.

CU-6	Asignar Permiso a Grupo.
Actor	Administrador.
Descripción	Permite al administrador asignar un permiso a un grupo seleccionado por él.
Referencia	RF6.

Tabla 8: Listado de casos de uso: Asignar Permiso a Grupo.

CU-7	Asignar Usuario a Grupo.
Actor	Administrador.
Descripción	Permite al administrador asignar un usuario a un grupo seleccionado por él.
Referencia	RF7.

Tabla 9: Listado de casos de uso: Asignar Usuario a Grupo.

CU-8	Asignar Permiso a Usuario.
Actor	Administrador.
Descripción	Permite al administrador asignar a un usuario seleccionado un permiso.
Referencia	RF6.

Tabla 10: Listado de casos de uso: Asignar Permiso a Usuario.

CU-9	Gestionar Acción.
Actor	Administrador.
Descripción	Permite al administrador Adicionar, Editar, Eliminar o Mostrar una acción.
Referencia	RF9, RF9.1, RF9.2, RF9.3, RF9.4.

Tabla 11: Listado de casos de uso: Asignar Permiso a Usuario.

CU-10	Listar Acciones.
Actor	Administrador.
Descripción	Permite al administrador obtener un listado con las acciones del sistema.
Referencia	RF10.

Tabla 12: Listado de casos de uso: Listar Acciones.

CU-11	Gestionar Módulo.
Actor	Administrador.
Descripción	Permite al administrador Adicionar, Editar, Eliminar o Mostrar módulos del sistema.
Referencia	RF11, RF11.1, RF11.2, RF11.3, RF11.4.

Tabla 13: Listado de casos de uso: Gestionar Módulo.

CU-12	Listar Módulos.
Actor	Administrador.
Descripción	Permite al administrador obtener un listado con los módulos del sistema.
Referencia	RF12.

Tabla 14: Listado de casos de uso: Listar Módulos.

CU-13	Gestionar Local Tecnológico.
Actor	Administrador.
Descripción	Permite al administrador Adicionar, Eliminar o Mostrar un local tecnológico.
Referencia	RF13, RF13.1, RF13.3, RF13.4.

Tabla 15: Listado de casos de uso: Gestionar Local Tecnológico.

CU-14	Editar Local Tecnológico.
Actor	Administrador, Editor.
Descripción	El sistema muestra un listado con los locales tecnológicos, el usuario escoge el local tecnológico que desee modificar y actualiza los datos pertinentes.
Referencia	RF13.2.

Tabla 16: Listado de casos de uso: Editar Local Tecnológico.

CU-15	Listar Locales Tecnológicos
Actor	Usuario, Administrador, Editor.
Descripción	Permite a los usuarios mostrar un listado de los locales tecnológicos del sistema
Referencia	RF14.

Tabla 17: Listado de casos de uso: Listar Locales Tecnológicos.

CU-16	Asignar Estaciones de Trabajo a Local Tecnológico
Actor	Usuario, Administrador, Editor.
Descripción	Permite asignar a un local tecnológico seleccionado las estaciones de trabajo que poseen su misma puerta de enlace.
Referencia	RF15.

Tabla 18: Listado de casos de uso: Asignar Estaciones de Trabajo a Local Tecnológico.

CU-17	Gestionar Tipo de Local.
Actor	Administrador.
Descripción	Permite al administrador Adicionar, Editar, Eliminar o Mostrar un tipo de local tecnológicos del sistema.
Referencia	RF16, RF16.1, RF16.2, RF16.3, RF16.4.

Tabla 19: Listado de casos de uso: Gestionar Tipo de Local.

CU-18	Listar Tipos de Locales.
Actor	Administrador.
Descripción	Permite al administrador obtener un listado con los tipos de locales del sistema.
Referencia	RF17.

Tabla 20: Listado de casos de uso: Listar Tipos de Locales.

CU-19	Consultar Estación de Trabajo.
Actor	Usuario, Administrador, Editor.
Descripción	Permite mostrar la información general, de hardware y software de una estación de trabajo, consultada de la base de datos de GRHS.
Referencia	RF18.

Tabla 21: Listado de casos de uso: Consultar Estación de Trabajo.

2.8. Diagramas de Casos de Uso del sistema.

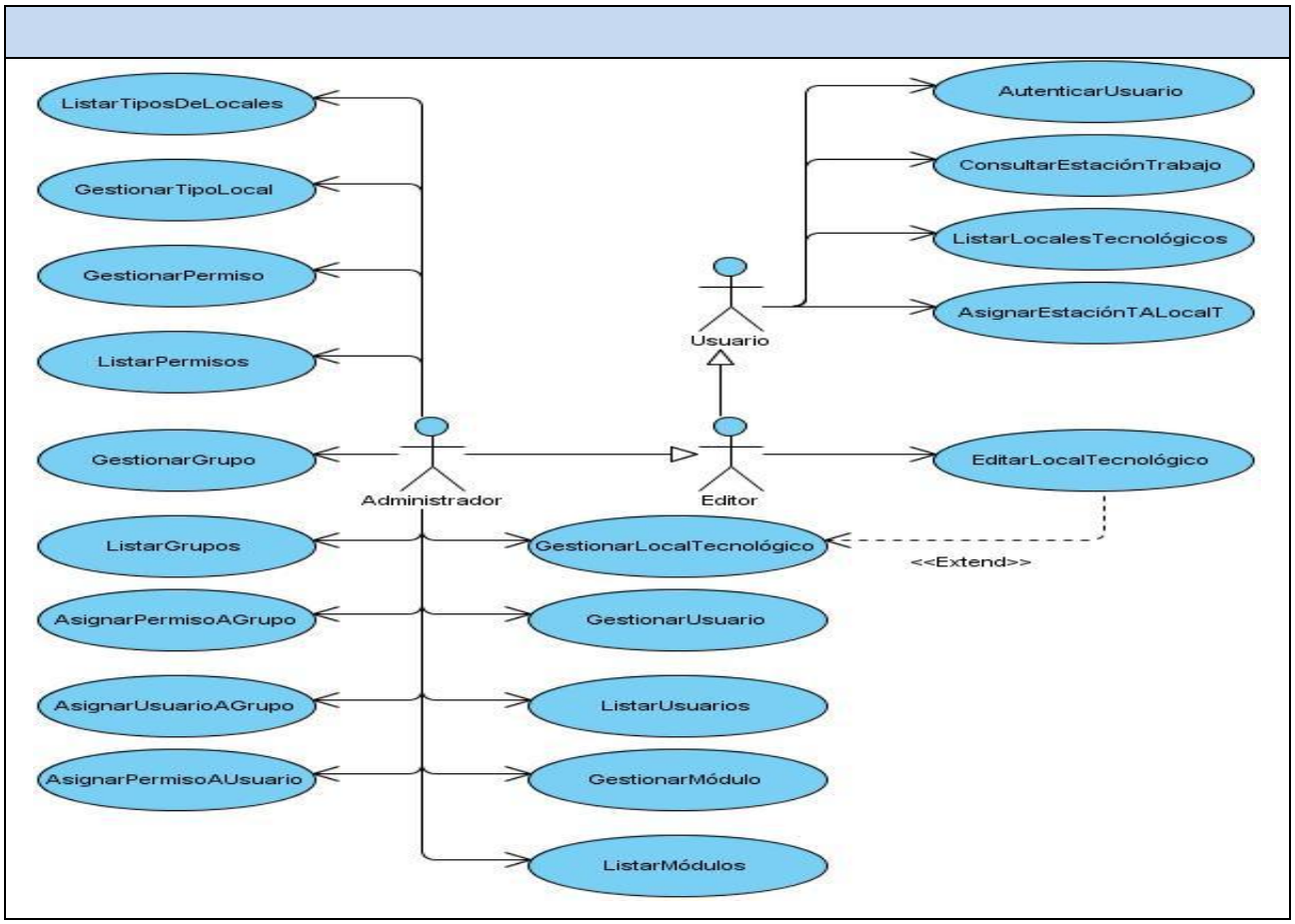

Figura 3: Diagrama de Casos de Uso.

2.9. Casos de Usos expandidos.

<table>
<tr><th colspan="3">Caso de uso</th></tr>
<tr><td>CU-13</td><td colspan="2">Gestionar Local Tecnológico.</td></tr>
<tr><td>Propósito</td><td colspan="2">Adicionar, Eliminar o Mostrar un local tecnológico, ya sea un laboratorio o una oficina.</td></tr>
<tr><td colspan="3">Actores: Administrador.</td></tr>
<tr><td colspan="3">Resumen: El caso de uso inicia cuando el administrador del sistema desea añadir, eliminar o mostrar un local tecnológico. El sistema permite introducir los datos del local y lo adiciona. También, posibilita eliminar un local seleccionado o mostrar sus datos.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF13, RF13.1, RF13.3, RF13.4.</td></tr>
<tr><td colspan="3">Flujo Básico</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>1. El caso de uso inicia cuando el actor accede a realizar una acción sobre un local tecnológico.</td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2">2. Brinda la posibilidad de realizar las acciones:
Adicionar Local.
Eliminar Local. Ver Sección 1: “Eliminar Local Tecnológico”.
Mostrar Local. Ver Sección 2: “Mostrar Local Tecnológico”.</td></tr>
<tr><td>3. Presiona el botón “Adicionar”.</td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2">4. El sistema muestra una interfaz con los campos necesarios para adicionar un nuevo local:
Denominación
Puerta de Enlace
Capacidad
Tipo de Local
Descripción
Permite adicionar el local.</td></tr>
<tr><td>5. Introduce los datos.
Presiona el botón “Guardar”.</td><td colspan="2"></td></tr>
</table>

	6. Valida los datos introducidos.
	7. Muestra un mensaje de confirmación. Permite repetir los pasos del 5 al 7 tantas veces como se desee. Terminando así el caso de uso.
Flujos Alternos	
*.a El actor presiona el botón “Cancelar”	
Acción del actor	**Respuesta del sistema**
	*.a.1 Vuelve a la vista principal. *.a.2 El caso de uso termina.
6. a Los datos introducidos son incorrectos o existe algún campo obligatorio vacío.	
Acción del actor	**Respuesta del sistema**
	6. a.1 Muestra un mensaje de error en cada campo incorrecto o vacío.
	6. a.2 Regresa al paso 5 del Flujo Básico.
6. b Los datos introducidos corresponden a un local existente.	
Acción del actor	**Respuesta del sistema**
	6. b.1 Muestra un mensaje de error indicando que existe un local con esas características.
	6. b.2 Vuelve a la vista principal.
Sección 1: “Eliminar Local Tecnológico”	
Flujo Básico	
Acción del actor	**Respuesta del sistema**
	1. Muestra un listado de los locales tecnológicos existentes.
2. Selecciona el local deseado. Presiona el botón “Eliminar”.	
	3. Muestra un mensaje de confirmación.
4. Presiona el botón “Sí”.	
	5. Elimina el local seleccionado y sus estaciones de trabajo. Actualiza el listado de locales tecnológicos. Terminando así el caso de uso.

Flujos Alternos	
4. a El usuario presiona el botón “No”.	
Acción del actor	**Respuesta del sistema**
	4. a.1 Vuelve a la vista principal.
	4. a.2 El caso de uso termina.
Sección 2: “Mostrar Local Tecnológico”	
Flujo Básico	
Acción del actor	**Respuesta del sistema**
	1. Muestra un listado de los locales tecnológicos existentes.
2. Selecciona el local deseado.	
	3. Muestra en la interfaz principal un formulario con los siguientes datos del local tecnológico: • Denominación • Puerta de Enlace • Capacidad • Descripción • Tipo Termina así el caso de uso.

Tabla 22: Caso de Uso Expandido Gestionar Local Tecnológico.

Caso de uso	
CU-14	Editar Local Tecnológico.
Propósito	Editar un local tecnológico, ya sea un laboratorio o una oficina.
Actores: Editor, Administrador.	
Resumen: El caso de uso inicia cuando el editor o el administrador del sistema, desea editar un local tecnológico. El sistema permite modificar los datos del local seleccionado.	
Referencias	RF13.2.
Flujo Básico	
Acción del actor	**Respuesta del sistema**
	1. Muestra un listado de los locales tecnológicos existentes.

2. Selecciona el local deseado. Presiona el botón “Editar”.	
	3. Muestra una interfaz con todos datos del local seleccionado: • Denominación • Puerta de Enlace • Capacidad • Descripción • Tipo Permite modificar el local.
4. Modifica los datos necesarios. Presiona el botón “Guardar”.	
	5. Valida los datos introducidos.
	6. Actualiza los datos del local. Permite repetir los pasos del 3 al 5 tantas veces como se desee. Terminando así el caso de uso.
Flujos Alternos	
*.a El actor presiona el botón “Cancelar”	
Acción del actor	**Respuesta del sistema**
	*.a.1 Vuelve a la vista principal. *.a.2 El caso de uso termina.
5. a Los datos modificados son incorrectos o existe algún campo obligatorio vacío.	
Acción del actor	**Respuesta del sistema**
	5. a.1 Muestra un mensaje de error en cada campo incorrecto o vacío.
	5. a.2 Regresa al paso 5 del Flujo Básico.
5. b Los datos modificados corresponden a un local existente.	
Acción del actor	**Respuesta del sistema**
	5. b.1 Muestra un mensaje de error indicando que existe un local con la misma denominación y/o puerta de enlace
	5. b.2 Vuelve a la vista principal.

Tabla 23: Editar Local Tecnológico.

<table>
<tr><th colspan="3">Caso de uso</th></tr>
<tr><td>CU-16</td><td colspan="2">Gestionar Tipo de Local Tecnológico.</td></tr>
<tr><td>Propósito</td><td colspan="2">Adicionar, Editar, Eliminar o Mostrar un tipo local tecnológico, ya sea laboratorio u oficina.</td></tr>
<tr><td colspan="3">Actores: Administrador.</td></tr>
<tr><td colspan="3">Resumen: El caso de uso inicia cuando el administrador del sistema desea añadir, editar, eliminar o mostrar un tipo de local tecnológico. El sistema permite introducir los datos del mismo y lo adiciona. Posibilita además, modificar o eliminar un tipo de local seleccionado o mostrar sus datos.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF13, RF13.1, RF13.3, RF13.4.</td></tr>
<tr><td colspan="3">Flujo Básico</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>1. El caso de uso inicia cuando el actor accede a realizar una acción sobre un tipo de local.</td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2">2. Brinda la posibilidad de realizar las acciones:
Adicionar Tipo de Local.
Editar Tipo de Local. Ver Sección 1: “Editar Tipo de Local”.
Eliminar Tipo de Local. Ver Sección 2: “Eliminar Tipo de Local”
Mostrar Tipo de Local. Ver Sección 3: “Mostrar Tipo de Local”.</td></tr>
<tr><td>3. Presiona el botón “Adicionar”.</td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2">4. El sistema muestra una interfaz con los campos necesarios para adicionar un nuevo tipo de local:
Denominación
Abreviatura
Descripción
Permite adicionar el local.</td></tr>
<tr><td>5. Introduce los datos.
Presiona el botón “Guardar”.</td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2">6. Valida los datos introducidos.</td></tr>
</table>

	7. Muestra un mensaje de confirmación. Permite repetir los pasos del 5 al 7 tantas veces como se desee. Terminando así el caso de uso.
Flujos Alternos	
*.a El actor presiona el botón “Cancelar”	
Acción del actor	**Respuesta del sistema**
	*.a.1 Vuelve a la vista principal. *.a.2 El caso de uso termina.
6. a Los datos introducidos son incorrectos o existe algún campo obligatorio vacío.	
Acción del actor	**Respuesta del sistema**
	6. a.1 Muestra un mensaje de error en cada campo incorrecto o vacío.
	6. a.2 Regresa al paso 5 del Flujo Básico.
6. b Los datos introducidos corresponden a un tipo de local existente.	
Acción del actor	**Respuesta del sistema**
	6. b.1 Muestra un mensaje de error indicando que existe ese tipo local.
	6. b.2 Vuelve a la vista principal.
Sección 1: “Editar Tipo de Local”	
Flujo Básico	
Acción del actor	**Respuesta del sistema**
	1. Muestra un listado con los tipos de locales existentes en el sistema.
2. Selecciona el tipo de local deseado. Presiona el botón “Editar”.	
	3. Muestra una interfaz con todos los datos del tipo de local seleccionado: • Denominación • Abreviatura • Descripción Permite modificar el tipo de local.
4. Modifica los datos necesarios.	

Presiona el botón “Guardar”.	
	5. Valida los datos introducidos.
	6. Actualiza los datos del tipo de local. Permite repetir los pasos del 3 al 5 tantas veces como se desee. Terminando así el caso de uso.
Flujos Alternos	
*.a El actor presiona el botón “Cancelar”	
Acción del actor	**Respuesta del sistema**
	*.a.1 Vuelve a la vista principal. *.a.2 El caso de uso termina.
5. a Los datos modificados son incorrectos o existe algún campo obligatorio vacío.	
Acción del actor	**Respuesta del sistema**
	5. a.1 Muestra un mensaje de error en cada campo incorrecto o vacío.
	5. a.2 Regresa al paso 5 del Flujo Básico.
6. b Los datos modificados corresponden a un tipo de local existente.	
Acción del actor	**Respuesta del sistema**
	5. b.1 Muestra un mensaje de error indicando que existe un local con la misma denominación y/o puerta de enlace
	5. b.2 Vuelve a la vista principal.
Sección 2: “Eliminar Tipo de Local”	
Flujo Básico	
Acción del actor	**Respuesta del sistema**
	1. Muestra un listado con los tipos de locales existentes en el sistema.
2. Selecciona el tipo de local deseado. Presiona el botón “Eliminar”.	
	3. Muestra un mensaje de confirmación.
4. Presiona el botón “Sí”.	
	5. Elimina el tipo de local seleccionado. Actualiza el listado de tipos de locales. Terminando

	así el caso de uso.
Flujos Alternos	
4. a El usuario presiona el botón “No”.	
Acción del actor	**Respuesta del sistema**
	4. a.1 Vuelve a la vista principal.
	4. a.2 El caso de uso termina.
Sección 3: “Mostrar Tipo de Local”	
Flujo Básico	
Acción del actor	**Respuesta del sistema**
1. Selecciona el local deseado.	
	2. Muestra en la interfaz un formulario con los siguientes datos del local tecnológico: • Denominación • Abreviatura • Descripción Termina así el caso de uso.

Tabla 24: Gestionar Tipo de Local Tecnológico.

2.10. Conclusiones parciales.

En este capítulo quedaron fundamentadas las características que deberá tener el sistema según las necesidades del proceso que se necesita informatizar. Se realizó además la propuesta de solución, obteniéndose a partir de un análisis las funcionalidades que debe realizar el sistema, las cuales se representaron mediante el Diagrama de Casos de Uso. Con estos resultados, se puede comenzar a construir el sistema, dándole cumplimiento a los requerimientos establecidos en este capítulo.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1. Introducción.

RUP define el análisis y el diseño, como un único flujo de trabajo en el que se realizan desde el inicio del desarrollo del software. El presente capítulo está dedicado a este flujo de trabajo, donde se traducen los requisitos funcionales a una especificación que describe cómo implementar la solución propuesta. En el capítulo se define la arquitectura seleccionada y los patrones de diseño que serán utilizados en la implementación de la solución. Quedan definidos los diagramas de clases del análisis especificando qué clases toman parte del caso de uso y las relaciones entre ellas. Además se muestran los diagramas de clases del diseño que describen la realización física de los casos de uso y los diagramas de secuencia como una descripción gráfica de la interacción entre los actores y el sistema. Se estructura la información que persistirá a través del diseño de la base de datos y los Diagramas de Entidad-Relación con las descripciones de algunas de sus tablas. En otro punto se especifica cómo será efectuado el tratamiento de los errores. Se plantea por otra parte, el proceso de la seguridad en el sistema. La representación de la interfaz es otro de los aspectos tratados en el capítulo y por último se muestra la concepción del Manual de Usuario para el sistema.

3.2. Modelo de análisis.

El modelo de análisis RUP lo define como la vista interna del sistema descrita en el lenguaje del desarrollador. Es utilizado fundamentalmente para comprender como diseñar e implementar el sistema. Además define las realizaciones de casos de uso. El objetivo del modelo de análisis es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y ayude a estructurar el sistema, incluyendo su arquitectura. (21)

3.2.1. Diagrama de clases del análisis.

El diagrama de clases del análisis se realiza para cada caso de uso del sistema y muestra las clases participantes, y sus relaciones. Es un artefacto en el que se representan los conceptos en un dominio del problema y se identifican tres tipos de clases: interfaz, controladoras y entidades.

Las clases **interfaz** se utilizan para modelar la interacción entre el sistema y sus actores, es decir, usuarios y sistemas externos. Cada clase debe asociarse con al menos un actor.

Las clases de **control** representan coordinación, secuencia, transacciones y control de otros objetos. Se usan para encapsular el control de un caso de uso en concreto y representar derivaciones y cálculos complejos, como la lógica del negocio. Los aspectos dinámicos del sistema

se modelan con estas clases, debido a que ellas manejan y coordinan las acciones y los flujos de control principales, y delegan trabajo a otros objetos, es decir, objetos de interfaz y de entidad.

Las clases **entidad** se utilizan para modelar la información que posee una larga vida y que es a menudo persistente. Modelan el comportamiento asociado a algún fenómeno o concepto, como una persona, un objeto o suceso del mundo real. Estas clases suelen mostrar una estructura de datos lógica y contribuyen a comprender de qué información depende el sistema.

A continuación se muestran el diagrama de clases del análisis de los Módulos Local y Tipo de Local:

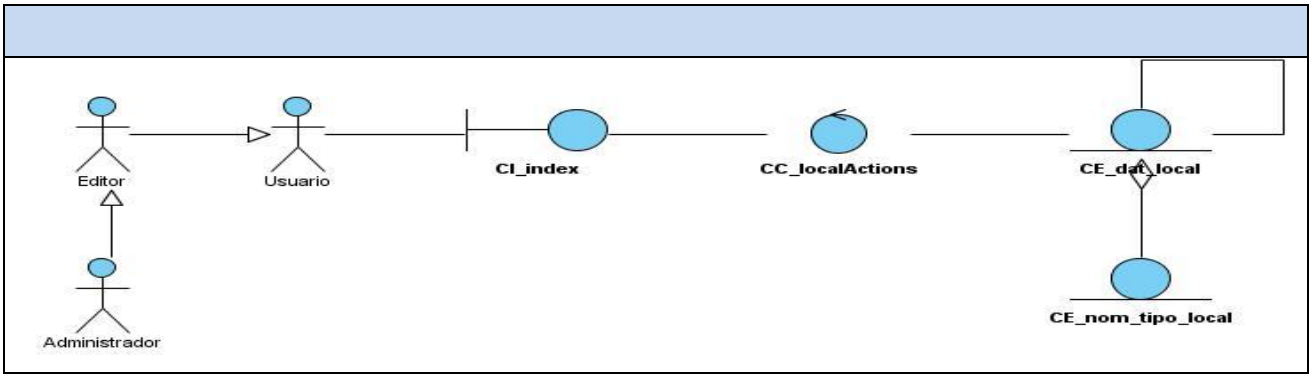

Figura 4: Diagrama de Clases del Análisis. Módulo Local.

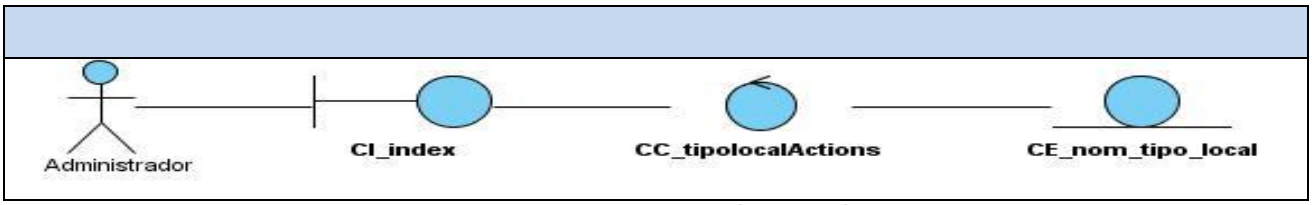

Figura 5: Diagrama de Clases del Análisis. Módulo Tipo Local.

3.3. Arquitectura

Modelo-Vista-Controlador (MVC).

En esta arquitectura el modelo, las vistas y los controladores se tratan como entidades separadas; provocando que cualquier cambio producido en el modelo se refleje automáticamente en cada una de las vistas. La arquitectura MVC presenta varias ventajas, entre las que cabe destacar:

- ✓ Una clara separación entre los componentes de un programa; que permite implementarlos por separado.
- ✓ Un API (Interfaz de Programación de Aplicaciones) muy bien definido; cualquiera que use el API, podrá reemplazar el modelo, la vista o el controlador, sin aparente dificultad.

- ✓ La conexión entre el modelo y sus vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas.

3.3.1.1. Implementación del MVC por Symfony.

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por tres niveles:

- ✓ El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- ✓ La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- ✓ El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

En la siguiente figura se muestra su funcionamiento:

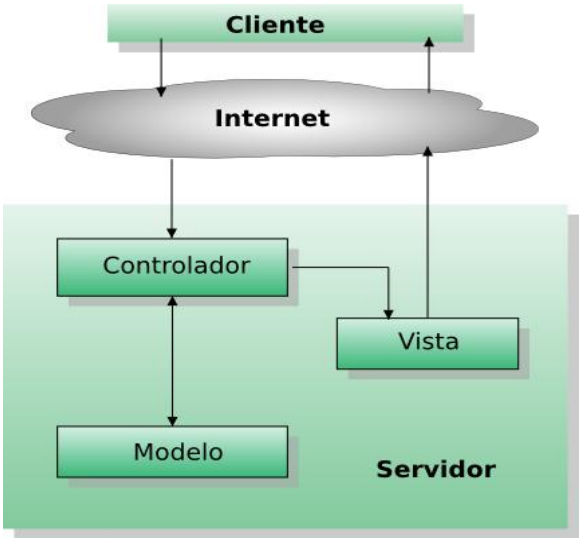

Figura 6: Esquema de la Arquitectura MVC.

La arquitectura del sistema está dividida en capas y sus niveles compuestos de la siguiente forma de acuerdo con la estructura del framework:

La capa de la Vista: Aprovecha la separación de código porque se descompone en:

- ✓ Layout: Es global en toda la aplicación.

- ✓ Lógica de la vista: Puede transformarse en un archivo de configuración sencillo, sin necesidad de programarla.
- ✓ Plantilla: Se encarga de visualizar las variables definidas en el controlador.

La capa del Controlador: Maneja las peticiones del usuario, la seguridad y carga la configuración de la aplicación. En esta capa de encuentra:

- ✓ Controlador frontal: Es único por cada aplicación y por tanto, el punto de entrada a la misma.
- ✓ Acciones: Incluyen el código específico del controlador en cada página.

La capa del Modelo: Esta capa se genera automáticamente, en función de la estructura de datos de la aplicación. Realiza el:

- ✓ Acceso a los datos.
- ✓ Abstracción de la base de datos. (18)

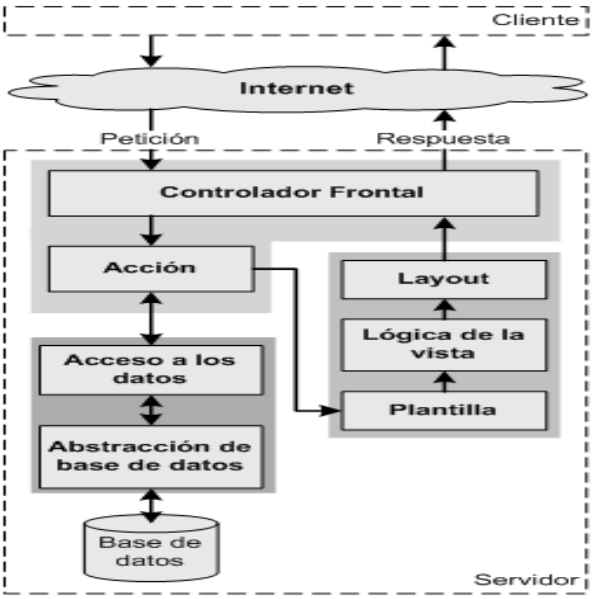

Figura 7: Flujo de trabajo del MVC en Symfony.

3.3.1.2. Organización de la aplicación.

Symfony organiza el código fuente en una estructura de tipo proyecto y almacena los archivos del proyecto en una estructura estandarizada de tipo árbol. Un proyecto se considera como un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio y que comparten el mismo modelo de objetos. Dentro de este, las operaciones se agrupan de forma lógica en aplicaciones y cada una está formada por uno o más módulos. Un módulo representa

una página web o un grupo de páginas con un propósito relacionado y en él se almacenan las acciones que representan las operaciones que se pueden realizar sobre un módulo. (18)

El sistema poseerá una estructura como la mostrada en la figura:

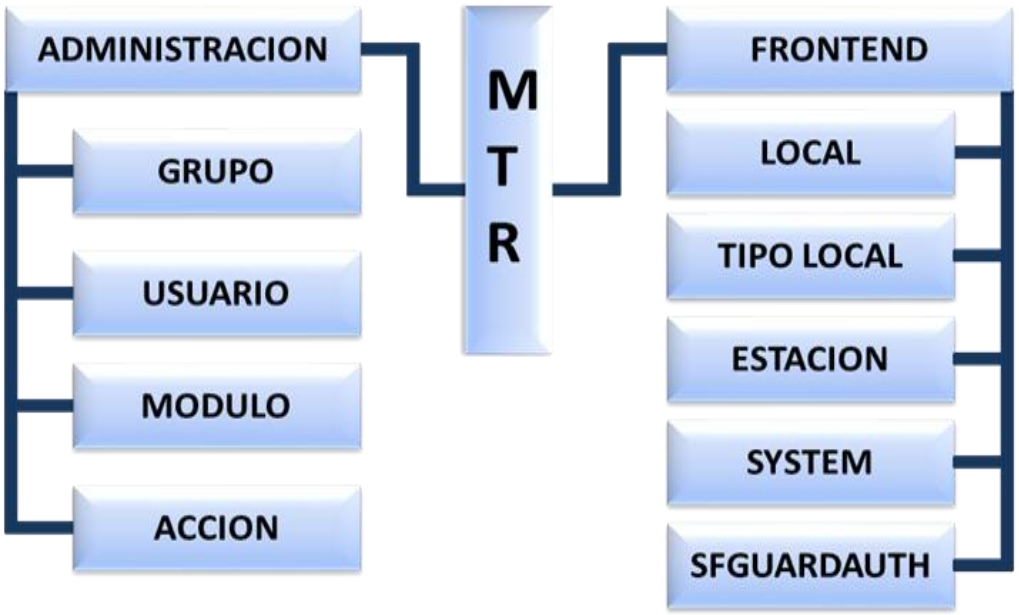

Figura 8: Organización de la aplicación.

3.4. Patrones de diseño.

Al usar un framework de PHP para desarrollar, se tiene que adoptar la metodología que impone, una de las principales ventajas a señalar en el uso de los framework es que están basados en patrones de diseño, siendo una de las características que hace posible la usabilidad que tienen. Symfony no queda exento de esto, y está concebido de tal manera que obliga al programador aplicar diferentes patrones de diseño. (22)

Patrones GRASP

Patrón Experto: Es uno de los patrones que más se utiliza cuando se trabaja con Symfony, con la inclusión de Doctrine para mapear la Base de Datos. Symfony utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular la lógica de los datos y generar las clases con las funcionalidades comunes de las entidades. Las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad y contienen la información necesaria de la tabla que representan.

Patrón Creador: En la clase Actions se encuentran las acciones definidas para el sistema. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que la clase Actions es "creador" de dichas entidades.

Bajo Acoplamiento: La clase Actions hereda únicamente de sfActions para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las vistas o el controlador, lo que proporciona que la dependencia en este caso sea baja.

Alta Cohesión: Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase Actions, la cual está formada por varias funcionalidades estrechamente relacionadas, siendo la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades.

Patrón Controlador: Un ejemplo del patrón Controlador es fácil de encontrar, por ejemplo en la clase sfFrontController, sfFrontWebController, sfContex, las "actions", el index.php del ambiente, entre otras. Symfony aplica el patrón "Front Controller" (Controlador frontal) y por tanto tiene una estructura organizada de controladores, que parte desde el "index.php" del ambiente y terminan en las "actions". En la capa del controlador cada clase tiene su responsabilidad y es única. Existen controladores que se encargan de la seguridad del sistema trabajando con ficheros YML, otros solo velan por identificar mediante datos las clases que deben realizar determinadas tareas.

Patrones GoF.

Patrón Decorator (Decorador): Este método pertenece a la clase abstracta sfView, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado layout.php es el que contiene el Layout de la página. Este archivo, conocido también como plantilla global, guarda el código HTML usual en todas las páginas para no tener que repetirlo en cada una. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla.

Singleton (Instancia única): Este patrón se aplica en el método getInstance de la clase sfRouting, el cual garantiza que esa clase sólo tenga una única instancia, proporcionando un punto de acceso global a la misma.

Patrón Command (Comando): Este patrón se observa en la clase sfWebFrontController, en el método dispatch (). Esta clase se genera automáticamente por Symfony y es la encargada de establecer el módulo y la acción que se va a ejecutar, según la petición del usuario. Este patrón se aplica además en la clase sfRouting, que está desactivada por defecto y procede según las necesidades del administrador del sistema donde se aplique el framework, se puede activar o desactivar

Patrón Registry (Registro): Este patrón es muy útil para los desarrolladores en la Programación Orientada a Objetos. Es un medio eficiente de compartir datos y objetos en la aplicación sin la necesidad de preocuparse por conservar numerosos parámetros o hacer uso de variables globales. Este patrón se aplica en la clase sfConfig, que es la encargada de acumular todas las variables de uso global en el sistema.

3.5. Modelo de diseño.

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen un impacto en el sistema a considerar. Sirve de abstracción de la implementación del sistema y es utilizado como una entrada fundamental de las actividades de implementación. Los propósitos que persigue el diseño son:

- ✓ Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación.
- ✓ Crear una entrada apropiada y un punto de partida para actividades de implementación capturando los requisitos, interfaces y clases. (21)

3.6. Diagrama de clases del diseño.

En el modelo de diseño, los casos de uso son realizados por las clases y los objetos del diseño, que se representan por colaboraciones y denota una realización de casos uso-diseño, esto ligado a algunas operaciones, atributos y asociaciones sobre una clase específica es manipulado por los diagramas de clases que conectados a una realización casos de uso y muestra las clases participantes y sus relaciones. Para realizar los diagramas de clases del diseño se usó la lógica con la que funciona el patrón MVC. De este modo las páginas cliente y los formularios constituyen las vistas, el controlador frontal y las clases actions las controladoras, y las clases de la lógica de negocio y el acceso a datos representan las modelos. (10)

A continuación se muestra el diagrama de clases del diseño del módulo local:

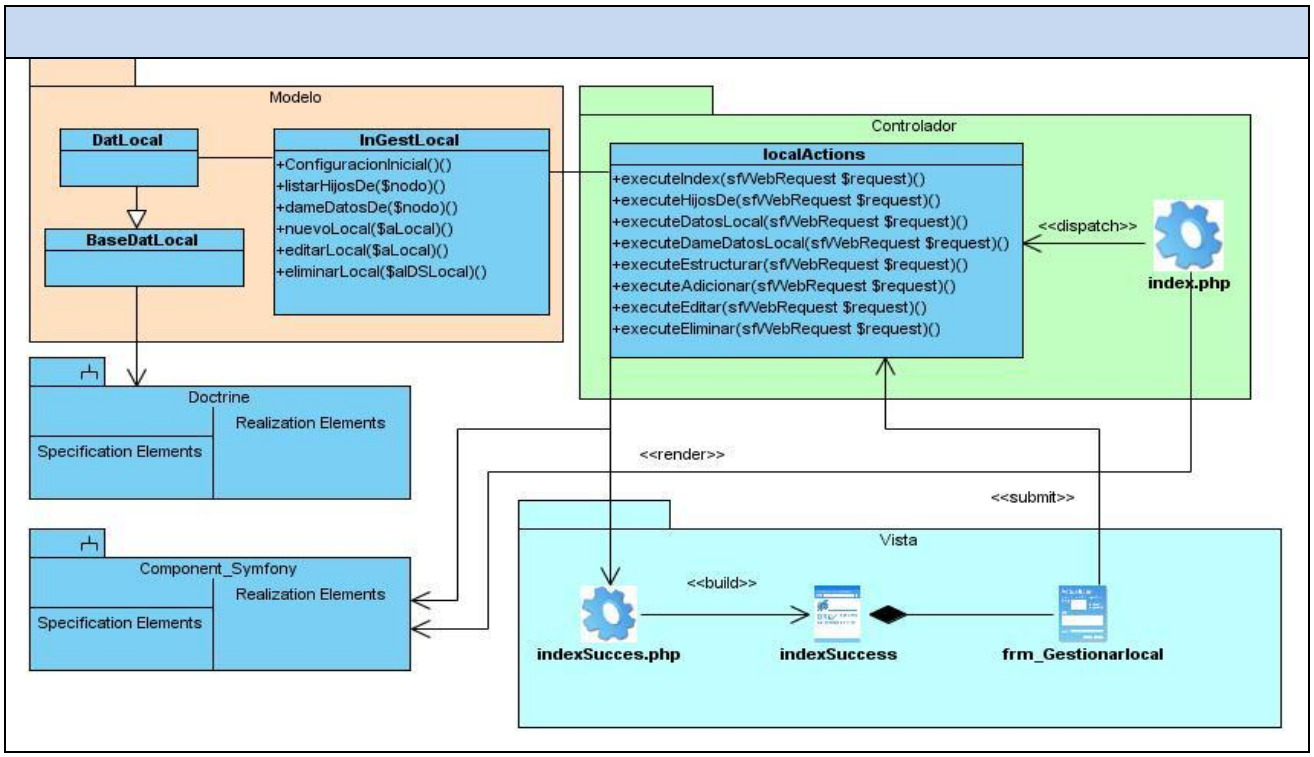

Figura 9: Diagrama de Clases del Diseño. Módulo Local.

A continuación se muestra la descripción de las clases del diseño del módulo local:

Nombre: localActions	
Tipo de clase:	Controladora
Para cada responsabilidad:	
Nombre:	executeIndex(sfWebRequest $request)
Descripción:	Método index por defecto del módulo local que ejecuta la vista del mismo.
Nombre:	executeHijosDe(sfWebRequest $request)
Descripción:	Método que devuelve los locales hijos de un local seleccionado.
Nombre:	executeDatosLocal(sfWebRequest $request)
Descripción:	Método que devuelve los datos de un local
Nombre:	executeAdicionar(sfWebRequest $request)
Descripción:	Método que permite adicionar un local, según los datos introducidos por el usuario.
Nombre:	executeEditar(sfWebRequest $request)

Descripción:	Método que permite modificar un local, según los datos introducidos por el usuario.
Nombre:	executeEliminar(sfWebRequest $request)
Descripción:	Método que permite eliminar un local, según los datos introducidos por el usuario.

Tabla 25: Descripción de la Clase localActions. Módulo Local.

Nombre: lnGestLocal	
Tipo de clase:	Modelo
Para cada responsabilidad:	
Nombre:	listarHijosDe($nodo)
Descripción:	Método que permite retornar los hijos de un módulo determinado.
Nombre:	dameDatosDe($nodo)
Descripción:	Método que permite retornar los datos de un local determinado.
Nombre:	nuevoLocal($aLocal)
Descripción:	Función para adicionar un local al sistema.
Nombre:	editarLocal($aLocal)
Descripción:	Función para editar un local del sistema.
Nombre:	eliminarLocal($aIDSLocal)
Descripción:	Función para eliminar un local del sistema.
Nombre:	cantidadLocales()
Descripción:	Función que retorna la cantidad de locales que hay en la base de datos.
Nombre:	existeLocal($local)
Descripción:	Función que retorna si existe o no un local dada su denominación.
Nombre:	existeMascara($mascara)
Descripción:	Función que retorna si existe o no un local dada su máscara.

Tabla 26: Descripción de la Clase lnGestLocal. Módulo Local.

A continuación se muestra el diagrama de clases del diseño del módulo tipo de local:

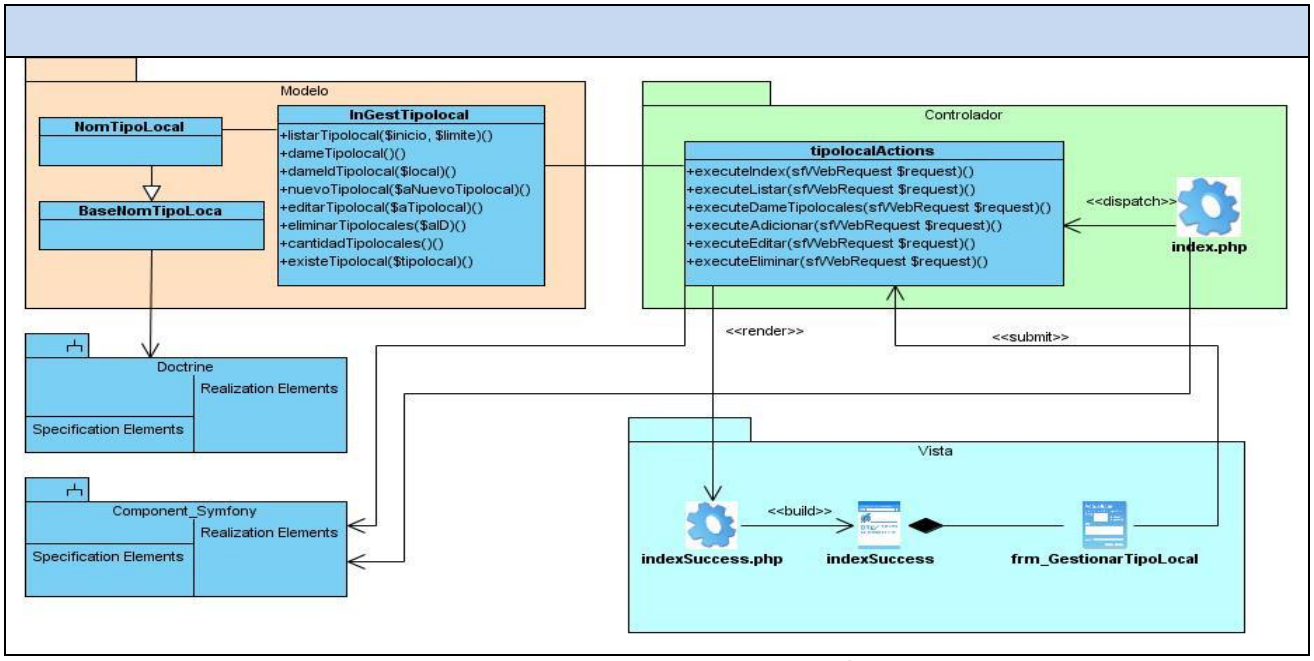

Figura 10: Diagrama de Clases del Diseño. Módulo Tipo Local.

A continuación se muestra la descripción de las clases del diseño del módulo tipo de local:

Nombre: tipolocalActions	
Tipo de clase:	Controladora
Para cada responsabilidad:	
Nombre:	executeIndex(sfWebRequest $request)
Descripción:	Método index por defecto del módulo tipo de local que ejecuta la vista del mismo.
Nombre:	executeListar(sfWebRequest $request)
Descripción:	Método que devuelve un tipo de local del sistema. Se usa para modificar el tipo de local.
Nombre:	executeDameTipolocales(sfWebRequest $request)
Descripción:	Método que devuelve los tipos de locales del sistema.
Nombre:	executeAdicionar(sfWebRequest $request)
Descripción:	Método que permite adicionar un tipo local, según los datos introducidos por el usuario.
Nombre:	executeEditar(sfWebRequest $request)

Descripción:	Método que permite modificar un tipo de local, según los datos introducidos por el usuario.
Nombre:	executeEliminar(sfWebRequest $request)
Descripción:	Método que permite eliminar un tipo de local seleccionado por el usuario.

Tabla 27: Descripción de la Clase tipolocalActions. Módulo Tipo Local.

Nombre: lnGestTipolocal	
Tipo de clase:	Modelo
Para cada responsabilidad:	
Nombre:	listarTipolocal($inicio, $limite)
Descripción:	Método que retorna un tipo de local del sistema. Se usa para modificar el tipo de local.
Nombre:	dameTipolocal()
Descripción:	Función que devuelve los tipos de locales del sistema.
Nombre:	dameIdTipolocal($local)
Descripción:	Función que retorna el identificador de un tipo de local determinado.
Nombre:	nuevoTipolocal($aNuevoTipolocal)
Descripción:	Función para adicionar un tipo de local al sistema.
Nombre:	editarTipolocal($aTipolocal)
Descripción:	Función para editar un tipo de local del sistema.
Nombre:	eliminarTipolocales($aID)
Descripción:	Función para eliminar un tipo de local del sistema.
Nombre:	cantidadTipolocales()
Descripción:	Función que retorna la cantidad de tipos de locales existentes en la base de datos.
Nombre:	existeTipolocal($tipolocal)
Descripción:	Función que retorna si existe o no, en la base de datos, un tipo de local determinado.

Tabla 28: Descripción de la Clase lnGestTipolocal. Módulo Tipo Local.

3.7. Diagramas de Interacción.

Se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva crear instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto a los mensajes enviados entre ellos. En el contexto de las clases se describe la forma en que los grupos de objetos colaboran para proveer un comportamiento. Mientras que un diagrama de casos de uso presenta una visión externa del sistema, la funcionalidad de dichos casos de uso se recoge como un flujo de eventos utilizando para ello interacciones entre sociedades de objetos. (10)

3.7.1. Diagramas de Secuencia.

Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes, diferenciándose del diagrama de colaboración, en el que se destaca la organización estructural de los objetos que envían y reciben mensajes.

A continuación se muestra el diagrama de secuencia del módulo local y sus escenarios:

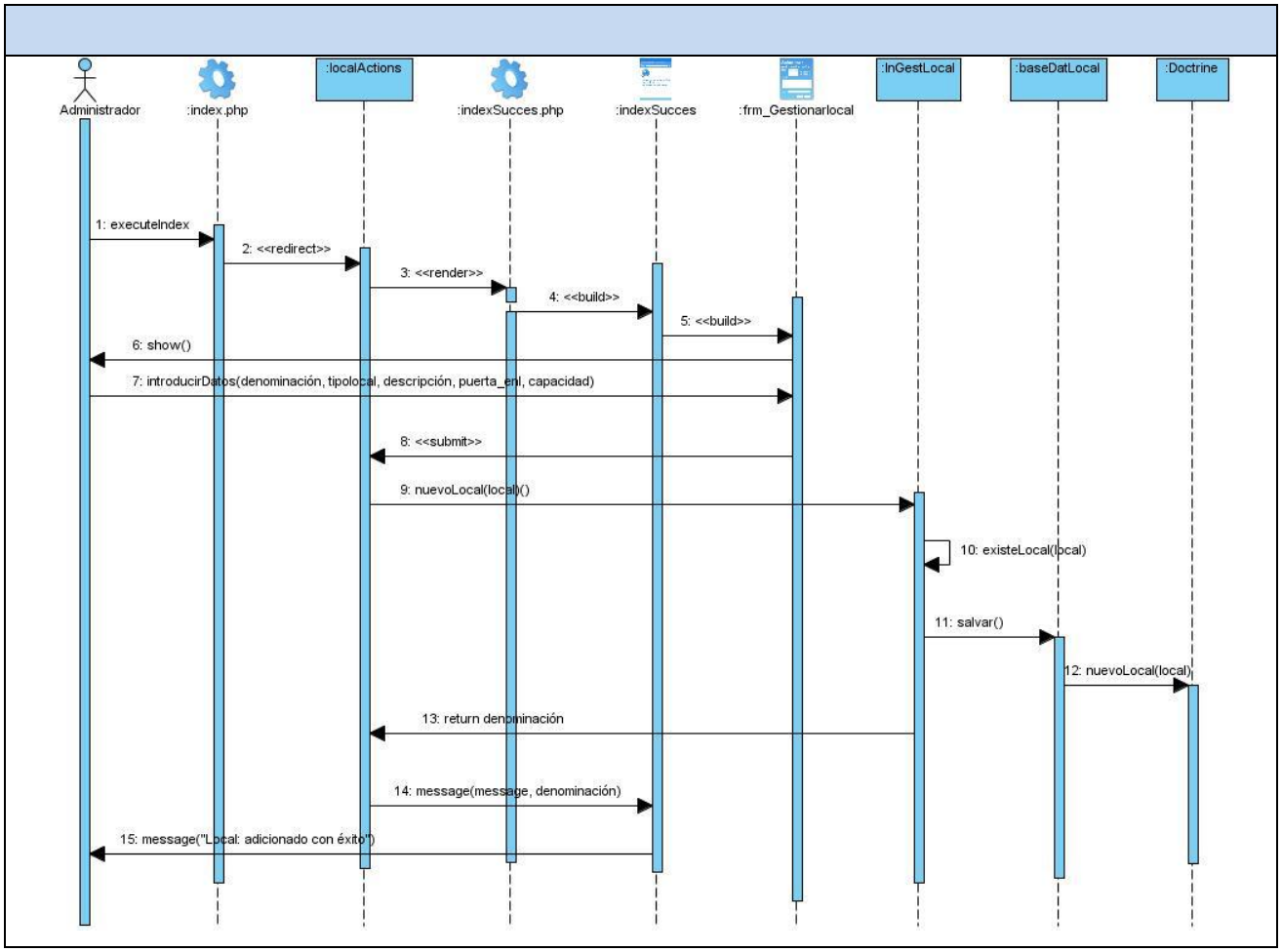

Figura 11: Diagrama de Secuencia. Módulo Local. Escenario Adicionar Local.

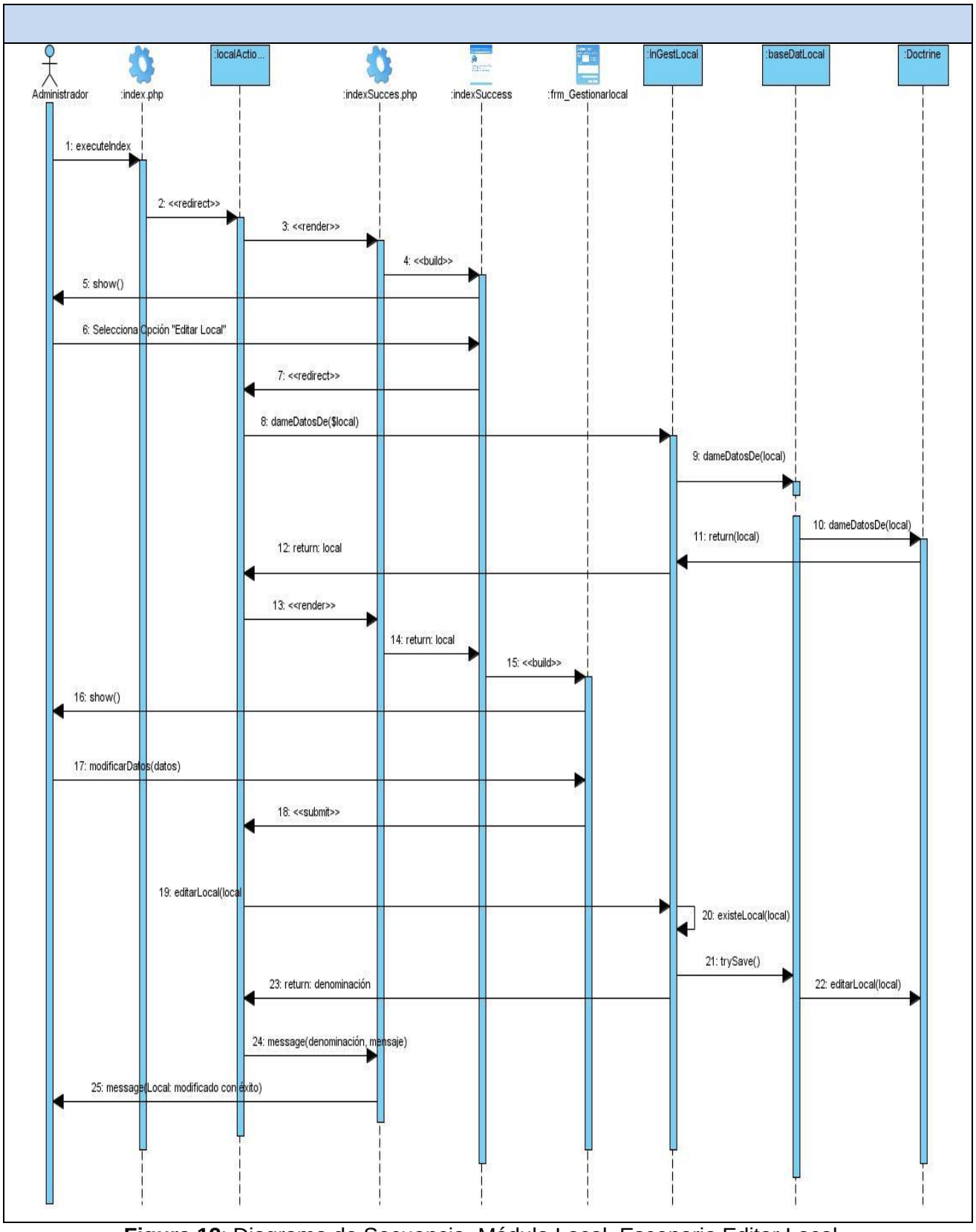

Figura 12: Diagrama de Secuencia. Módulo Local. Escenario Editar Local.

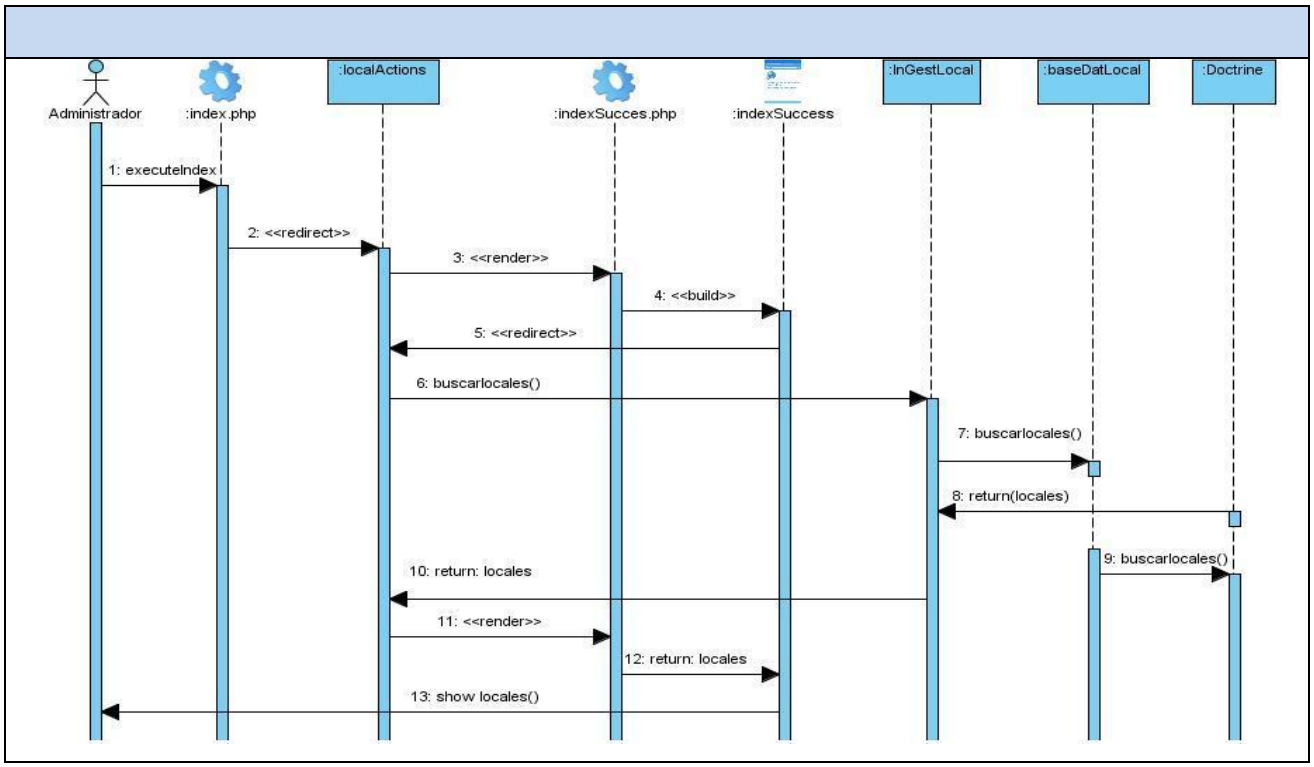

Figura 13: Diagrama de Secuencia. Módulo Local. Escenario Mostrar Local.

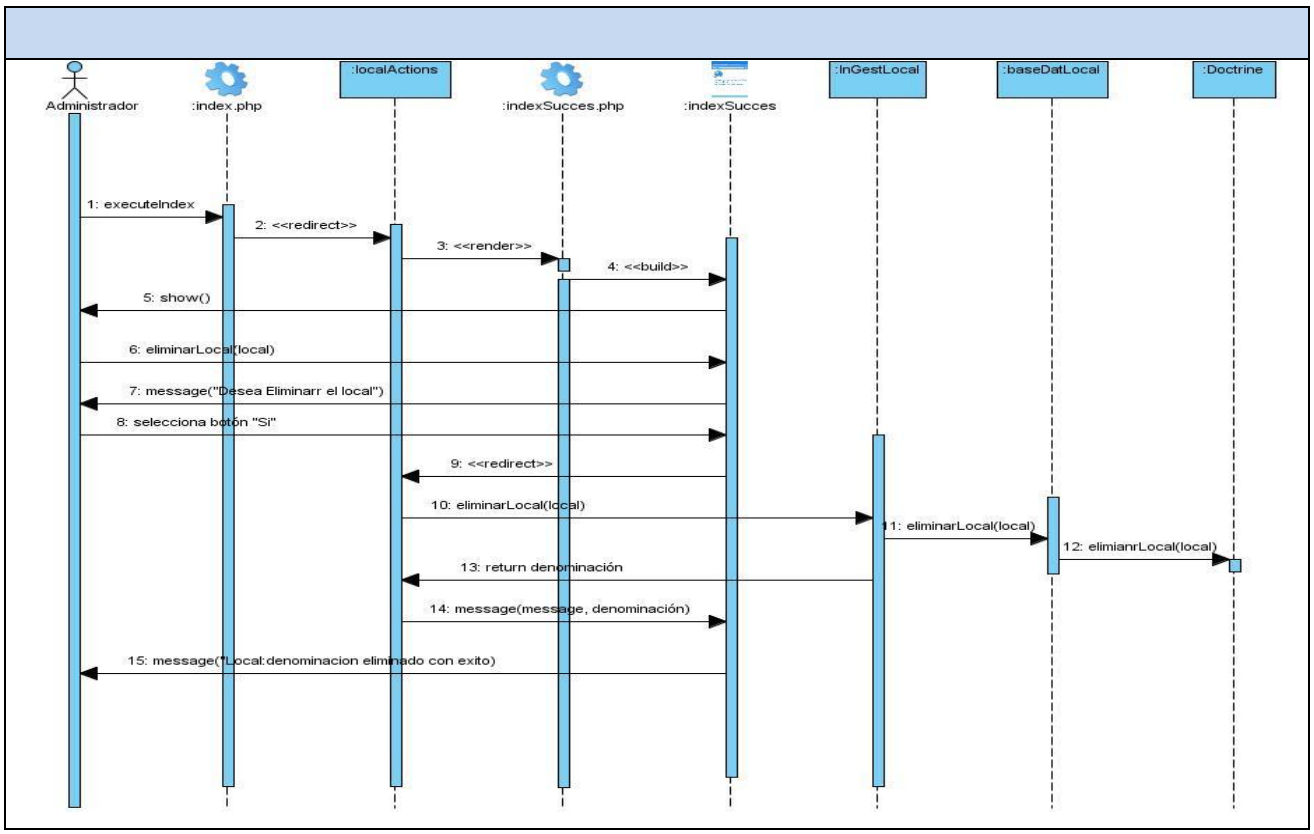

Figura 14: Diagrama de Secuencia. Módulo Local. Escenario Eliminar Local.

A continuación se muestra el diagrama de secuencia módulo tipo de local y sus escenarios:

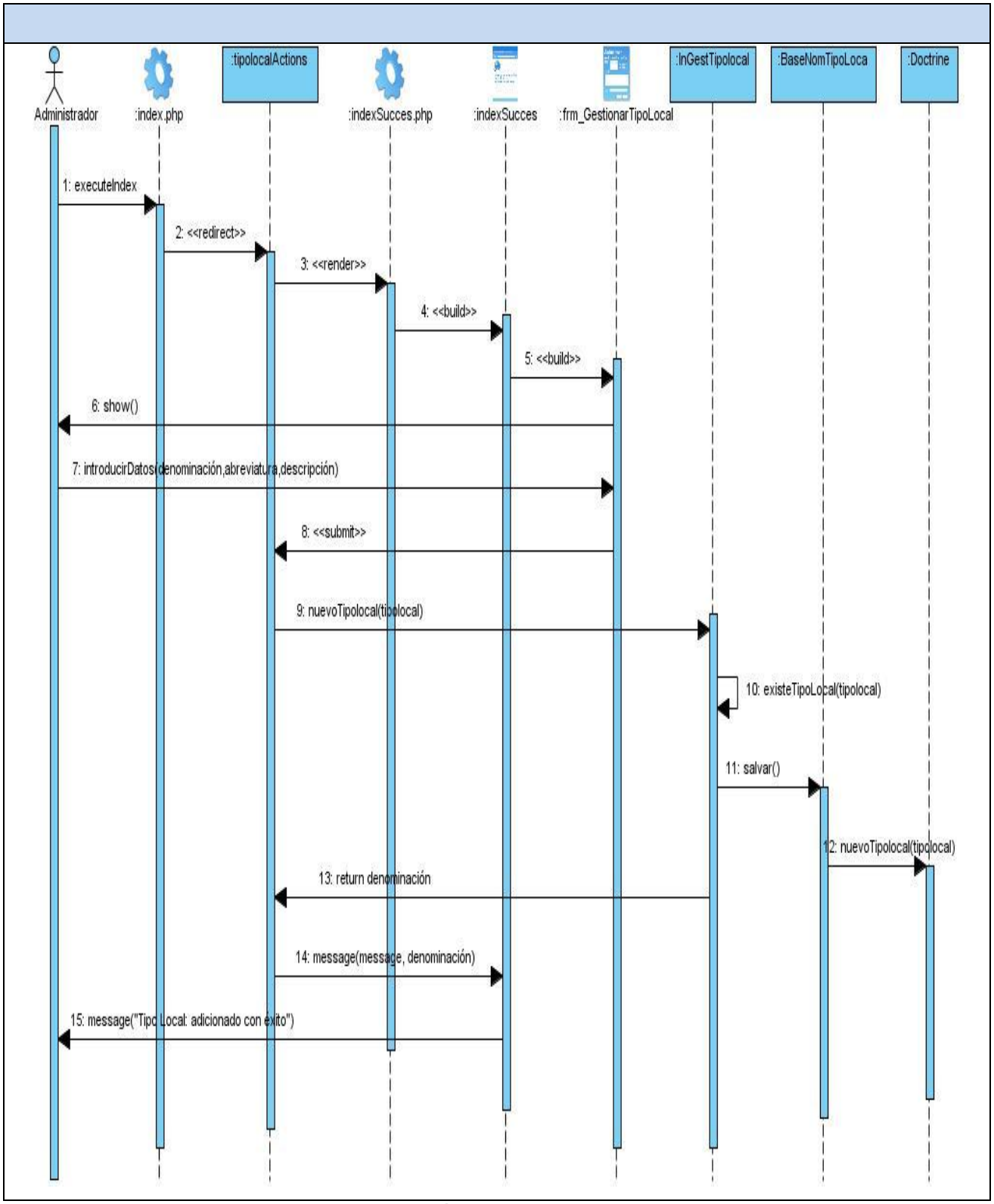

Figura 15: Diagrama de Secuencia. Módulo Tipo Local. Escenario Adicionar Tipo Local.

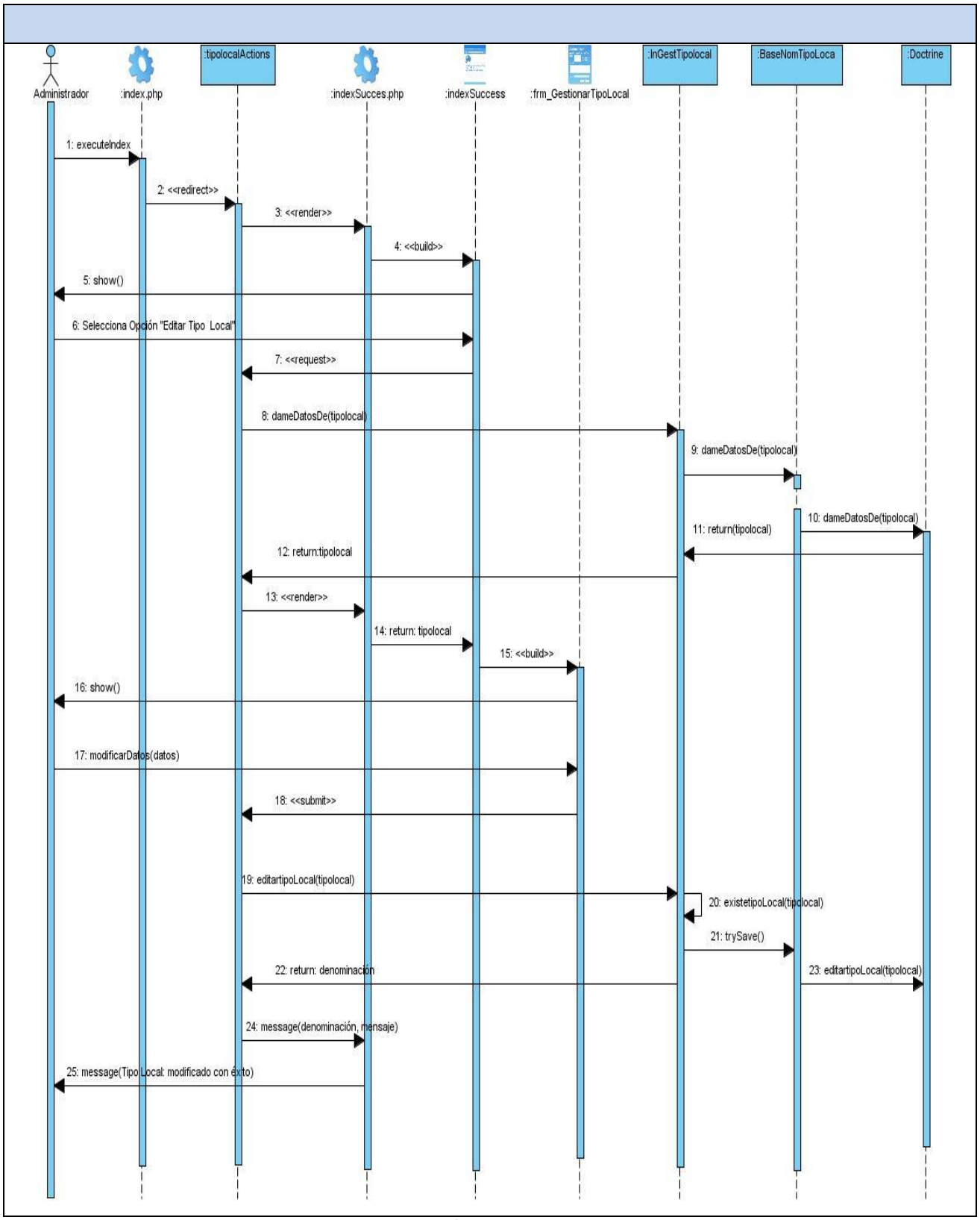

Figura 16: Diagrama de Secuencia. Módulo Tipo Local. Escenario Editar Tipo Local.

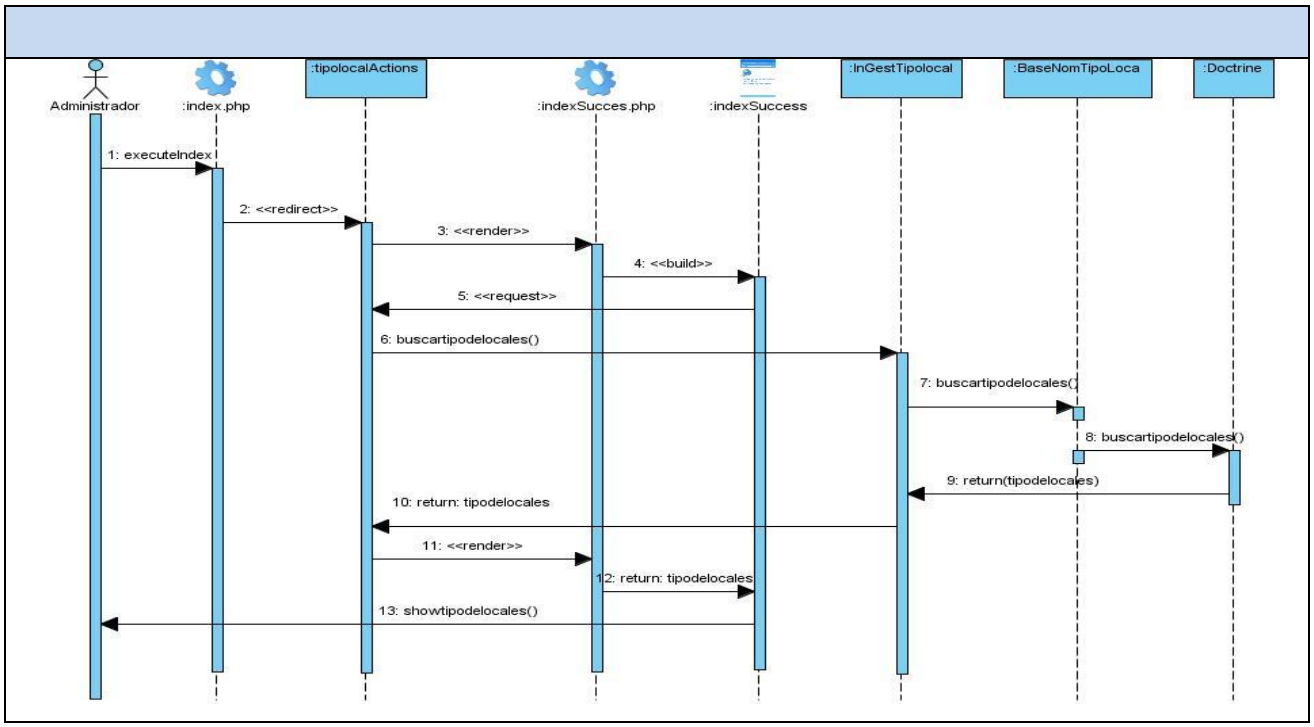

Figura 17: Diagrama de Secuencia. Módulo Tipo Local. Escenario Mostrar Tipo Local.

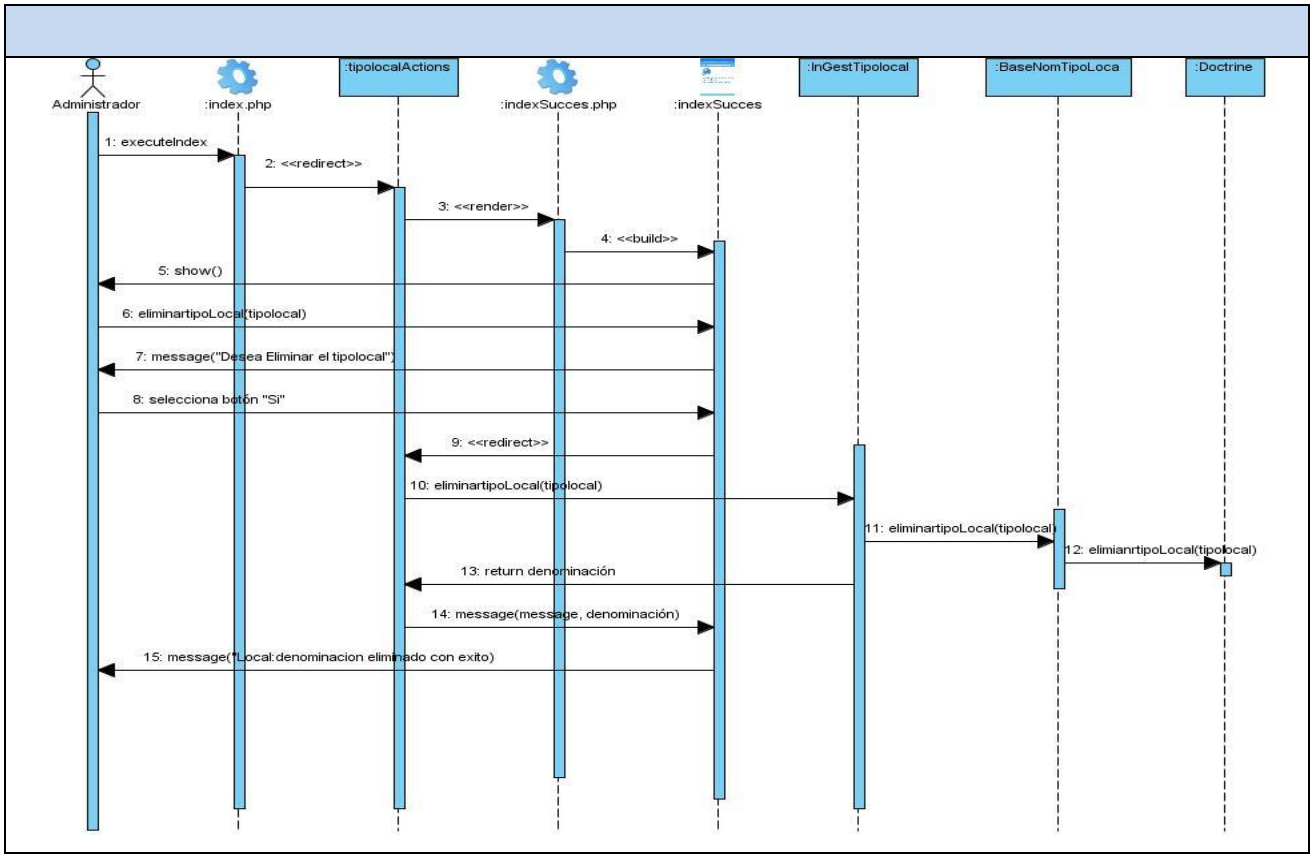

Figura 18: Diagrama de Secuencia. Módulo Tipo Local. Escenario Eliminar Tipo Local.

3.8. Diagrama de Clases Persistentes.

Todas las clases identificadas en el dominio del análisis no son persistentes. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Es responsabilidad del diseñador de definir las clases que deben ser persistentes.

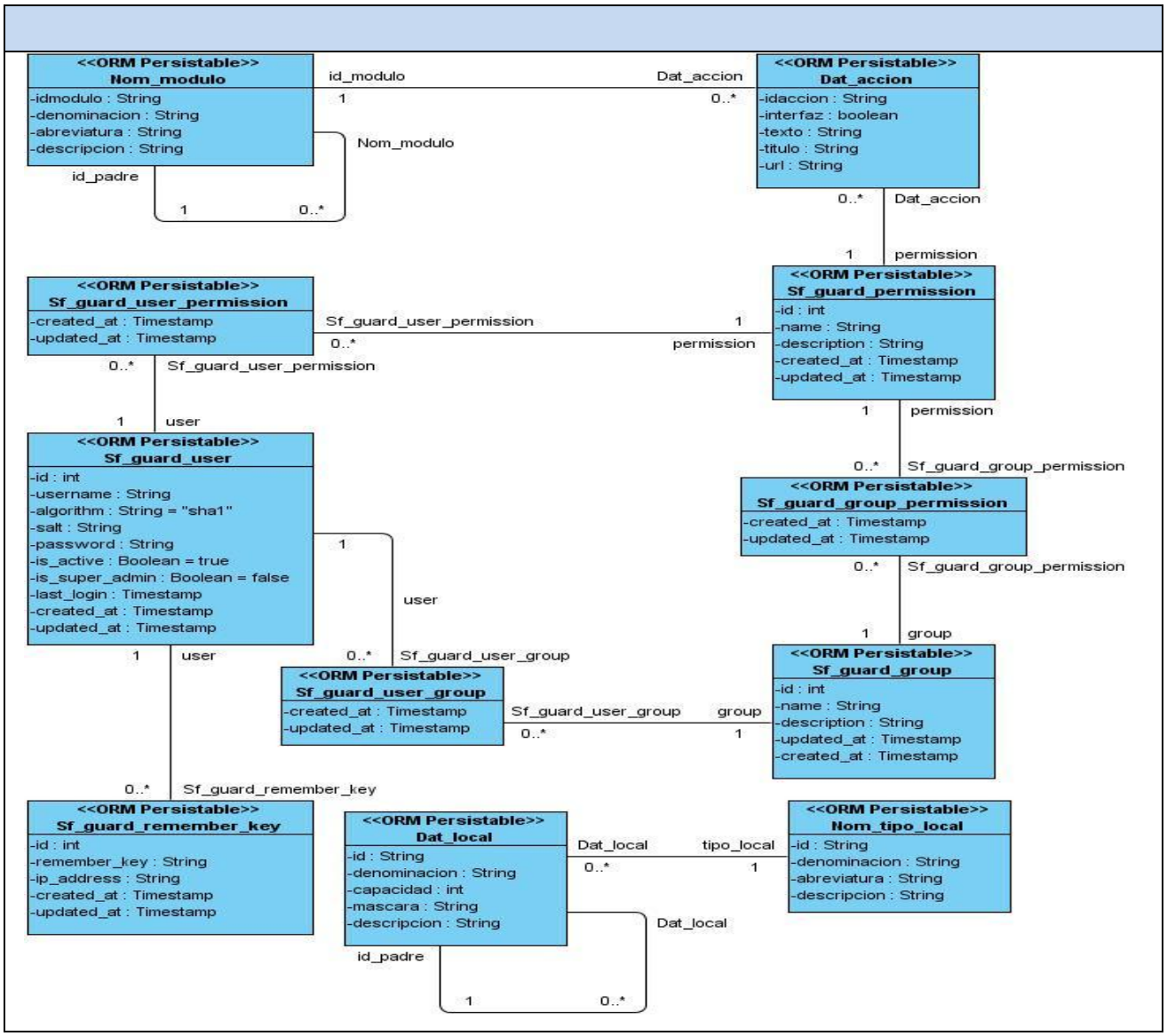

Figura 19: Diagrama de Clases Persistentes.

3.9. Modelo de Datos.

El objetivo de construir un Modelo de Datos es identificar y representar las tablas importantes para el funcionamiento del negocio (entidades), sus propiedades (atributos), y la forma en que las tablas se relacionan entre sí (relaciones).

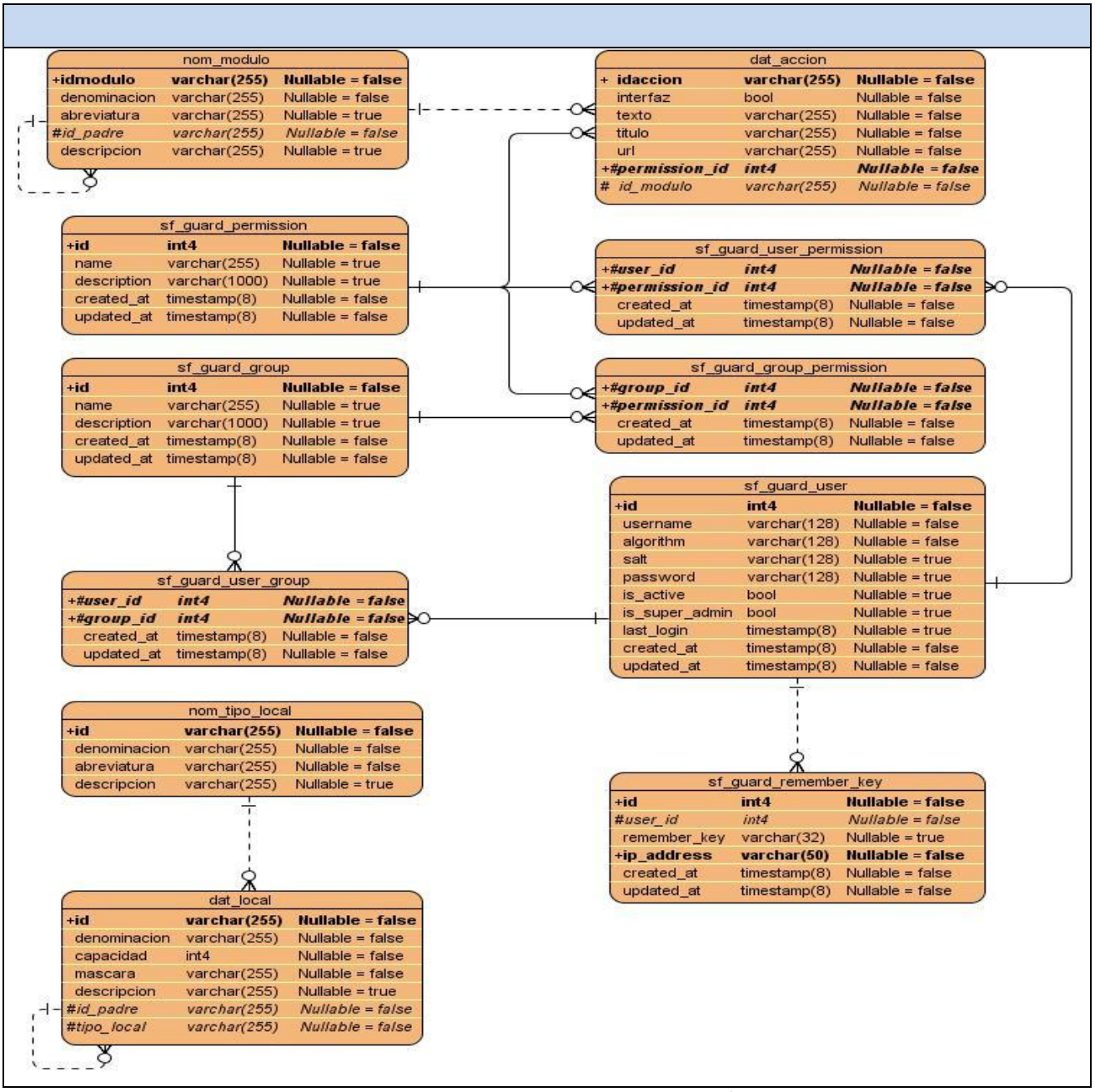

Figura 20: Diagrama Entidad-Relación.

3.9.1. Descripción de las tablas.

Nombre: Dat_local		
Descripción: En esta tabla se almacena la información de los locales del sistema.		
Atributo	**Tipo**	**Descripción**
id	String	Responde al identificador de un local tecnológico.
denominación	String	Responde al nombre de un local tecnológico.

abreviatura	String	Responde a la abreviatura de un local tecnológico.
capacidad	int	Responde a la capacidad de diseño de un local tecnológico.
puerta_enl	String	Responde a la puerta de enlace de un local
descripción	String	Responde a la descripción de un local.

Tabla 29: Descripción Tabla Dat_Local. Base de Datos Tesis.

Nombre: Nom_tipo_local		
Descripción: En esta tabla se almacena la información de los tipos de locales del sistema.		
Atributo	**Tipo**	**Descripción**
id	String	Responde al identificador de un tipo de local tecnológico.
denominación	String	Responde al nombre de un tipo de local tecnológico, dígase laboratorio de producción o docencia o una oficina.
abreviatura	String	Responde a la abreviatura de un tipo de local tecnológico.
descripción	String	Responde a la descripción de un tipo de local.

Tabla 30: Descripción Tabla Nom_tipo_local. Base de Datos Tesis.

3.10 Tratamiento de errores

En el sistema propuesto el tratamiento de errores se realiza con el fin de evitar, minimizar y tratar los posibles errores, garantizando de esta forma la integridad y confiabilidad de la información que se registra y se muestra al usuario. Antes de realizar cualquier acción se validan los datos y antes de eliminar se exige confirmación al usuario. El usuario es informado cuándo ha ejecutado o no, correctamente las acciones, por ejemplo al dejar campos requeridos en un formulario o al introducir datos no válidos. De forma general, siempre se notifica al usuario de los posibles errores tanto en la navegación como en la administración del sistema.

3.11 Seguridad

En el sistema la información se valida tanto en el cliente como en el servidor, no obstante los usuarios acceden de manera rápida al sistema sin que los requerimientos de seguridad se conviertan en un problema para ellos. A las contraseñas se les aplica un algoritmo de encriptación antes de almacenarlas en la base de datos. El controlador frontal ofrece un punto de entrada único para toda la aplicación, es el encargado de manejar la seguridad y sólo se puede ejecutar por defecto desde localhost. Los usuarios necesitan estar autenticados antes de realizar alguna funcionalidad o acceder a las partes de la aplicación.

3.12 Interfaz.

Las páginas de la interfaz del sistema están diseñadas con un estilo común garantizando que exista uniformidad y definiendo de forma única la manera de presentar los contenidos. Todo ello fue posible gracias al uso de hojas de estilo, lo que reduce el uso de imágenes que retarden el tiempo de respuesta de la aplicación. De igual modo la aplicación de administración brinda al usuario una interfaz amigable con menús desplegables y acceso a determinadas acciones por más de una vía, lo que sin duda mejora la usabilidad del sistema.

3.13 Concepción de la ayuda.

La “ayuda” es uno de los temas más sensibles para los usuarios a la hora de interactuar con el sistema, pues esta opción le permite conocer el funcionamiento del mismo. Esta debe estar accesible en todas las páginas de la aplicación, con el objetivo de que el usuario vea la información que necesita en ese momento. Consta con informaciones generales y específicas del sistema, así como explicaciones acerca de las funcionalidades del mismo.

3.14. Conclusiones parciales.

En este capítulo se mostraron los modelos del análisis y el diseño, en los cuales se modelaron los diagramas de clases del análisis y diseño respectivamente. Además, se presentó el diagrama de clases persistentes, obteniéndose el modelo de datos, lo que facilitó el diseño de las tablas de la base de datos. La generación de estos artefactos, la descripción de la arquitectura MVC que Symfony establece y los patrones de diseño en los que se basa, permitió obtener una mayor comprensión de la aplicación y definir los principios que guiarán la implementación y organización de la misma. Además se analizaron las estrategias a seguir para dar respuesta a los requisitos no funcionales relacionados con la seguridad, se obtuvo un prototipo de interfaz y las pautas de diseño para la misma, así como la concepción de la ayuda de la aplicación.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1. Introducción.

El presente capítulo se trata del flujo de trabajo de implementación y pruebas. En este flujo se construye el modelo necesario para desarrollar el proceso de implementación del sistema con los diagramas de componentes definidos. También se elabora el diagrama de despliegue donde se representan los nodos necesarios, en los que se distribuye la aplicación. Además se abordan los temas relacionados con las pruebas aplicadas al software.

4.2. Implementación.

4.2.1. Diagrama de Despliegue.

El diagrama de Despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Es un grafo de nodos unidos por conexiones de comunicación donde cada nodo puede contener instancias de componentes. En general un nodo puede ser una unidad de computación de algún tipo, desde un sensor hasta un mainframe y las instancias de componentes de software pueden estar unidas por relaciones de dependencia.

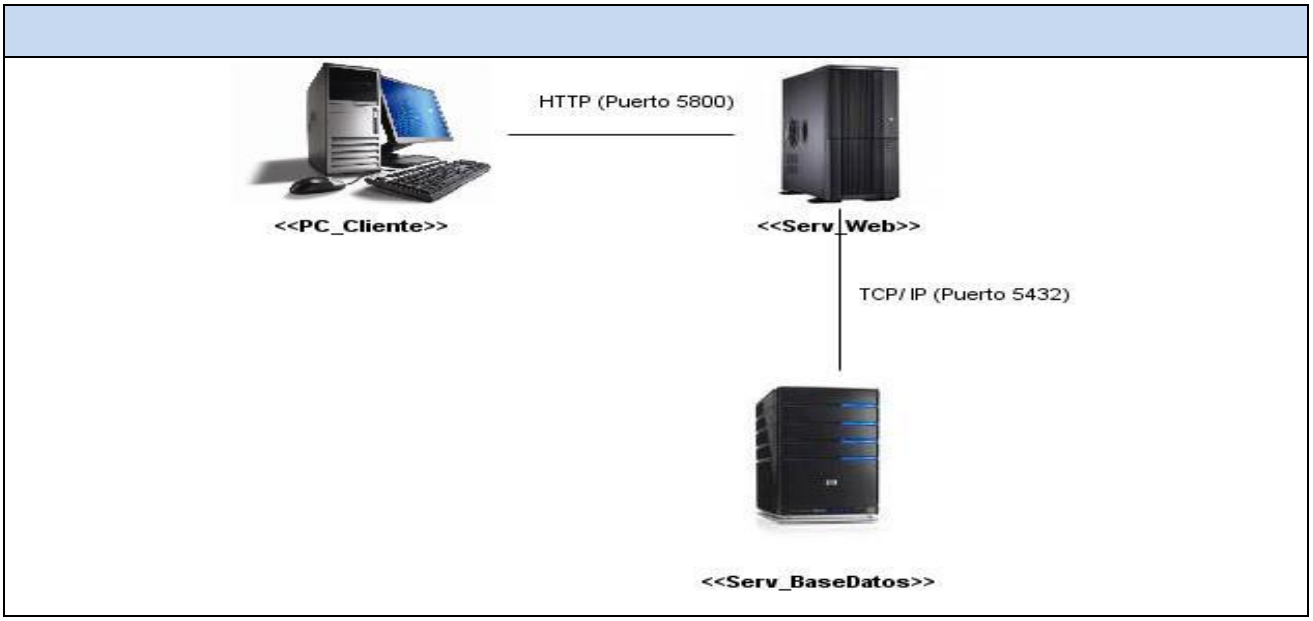

Figura 21: Diagrama de Despliegue.

4.2.2. Modelo de Implementación.

El modelo de implementación representa la composición física de la implementación en términos de subsistemas y elementos y describe cómo dichos elementos de diseño se implementan en

componentes. Se considera el artefacto más significativo del flujo de trabajo de implementación, debido a la importancia que tiene para los desarrolladores comprender el funcionamiento del sistema desde el punto de vista de componentes y sus relaciones. Este modelo está conformado por el diagrama de componentes.

4.2.2.1 Diagrama de Componentes.

Dentro del Modelo de Implementación se encuentran los diagramas de componentes. Un componente es la parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. Este diagrama describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros.

A continuación se muestra el diagrama de componentes del Módulo Local y Tipo de Local.

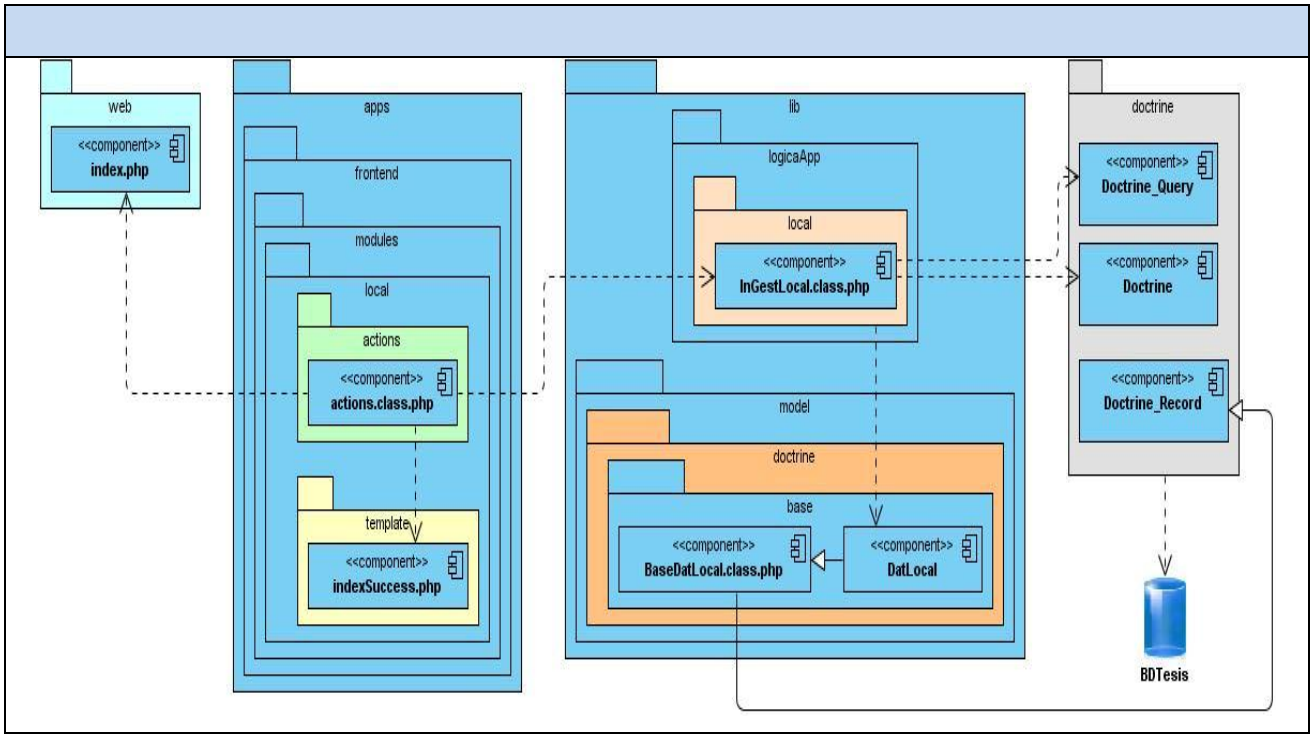

Figura 22: Diagrama de Componentes. Módulo Local.

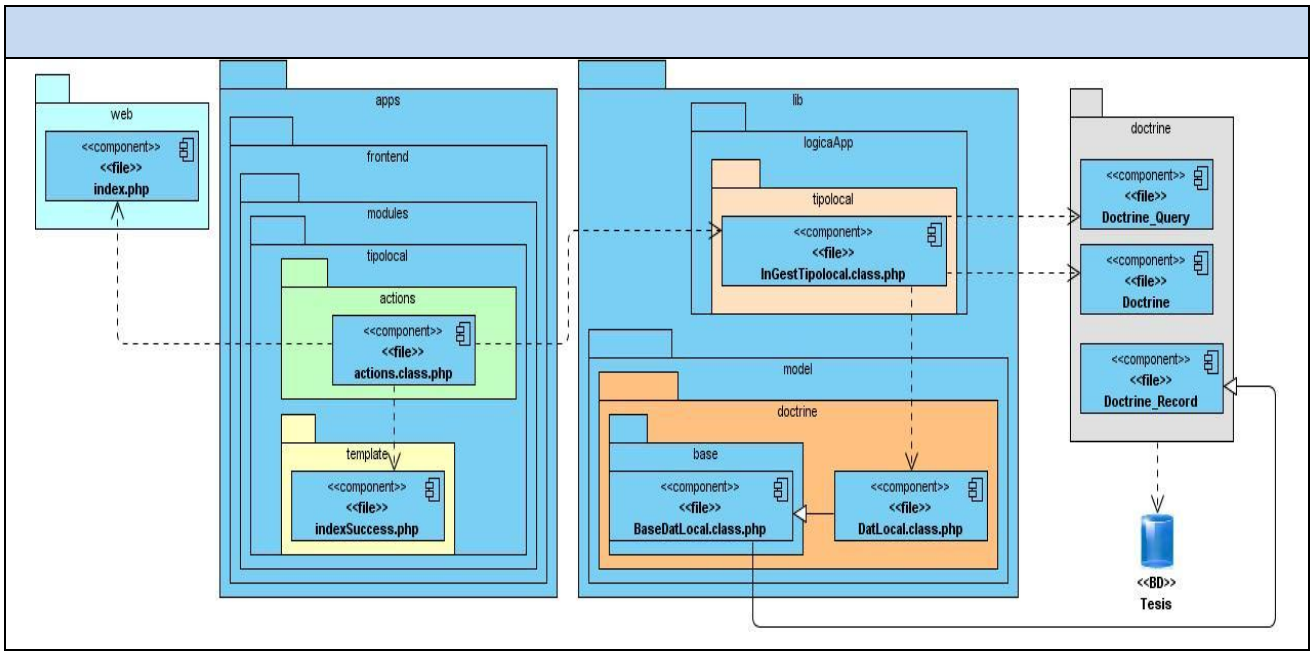

Figura 23: Diagrama de Componentes. Módulo Tipo Local.

4.3 Pruebas

Una de las últimas fases del ciclo de vida del desarrollo de software es el flujo de trabajo de pruebas, al cual es necesario dedicarle un importante tiempo, pues dicha actividad está encaminada a encontrar errores en el software.

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. (23)

Dicha actividad brinda varios beneficios. Por ejemplo la calidad es una variable muy importante para todo producto y uno de los caminos para garantizarla es siguiendo esta disciplina, además proporciona una medida del progreso del trabajo que se despliega.

4.3.1 Pruebas de caja negra.

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, es por ello que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, que se produce una salida correcta y que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo, fundamentalmente del sistema sin tener en cuenta, en gran medida, la estructura interna del software. (23)

4.3.2.1 Casos de pruebas.

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados, desarrollados para cumplir un objetivo en particular o una función esperada.

Los casos de pruebas deben verificar:

- ✓ Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificaciones de los requerimientos.
- ✓ Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño. (23)

Total de No Conformidades	# de No Conformidades que proceden	# de No Conformidades que no proceden	# de No Conformidades resueltas
8	7	1	7

Se realizaron dos iteraciones de pruebas de funcionalidad a la aplicación. En la primera iteración se encontraron un total de 8 no conformidades las cuales quedaron pendientes. En la segunda iteración se le dio solución a 7 no conformidades y una de ellas no procedió.

A continuación se muestran los casos de prueba de la segunda iteración de dos casos de uso críticos del sistema:

Prueba del caso de uso Gestionar Local escenario "Adicionar local".

Condición de entrada	Casos válidos	Casos no válidos
Denominación	Nombre del local	Campo vacío o local existente.
Puerta de Enlace	Puerta de enlace	Campo vacío, puerta de enlace incorrecta o existente.
Capacidad	Número entero.	Campo vacío o cadena de caracteres.
Tipo de Local	Seleccionar Tipo de local.	Campo vacío.

Caso de uso	Gestionar Local "Adicionar local".
Caso de prueba	Permitir adicionar un local en la base de datos introduciendo correctamente los datos del mismo.
Entrada	El usuario introduce los datos del local correctamente de la forma:

	Denominación: Lab_301 Puerta de Enlace: 10.31.16.255 Capacidad: 31 Tipo de Local: Laboratorios
Resultado	El sistema introduce el local en la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar Local "Adicionar local".**
Caso de prueba	Adicionar un local introduciendo incorrectamente los datos o dejando algún campo vacío.
Entrada	Existen campos vacíos o con datos incorrectos.
Resultado	El sistema no le permite al usuario entrar datos incorrectos en cada uno de los campos y muestra mensajes de error indicando que los mismos son obligatorios.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Caso de uso	**Gestionar Local "Adicionar local".**
Caso de prueba	Adicionar un local introduciendo una denominación o puerta de enlace de un local ya existente en la base de datos.
Entrada	El usuario introduce una denominación o puerta de enlace de un local ya existente en la base de datos.
Resultado	El sistema muestra un mensaje de error indicando que ya existe un local con igual denominación y/o puerta de enlace.
Condiciones	Debe existir una denominación o una puerta de enlace con igual nombre en la base de datos.

Prueba del caso de uso Gestionar Local escenario "Editar local".

Condición de entrada	**Casos válidos**	**Casos no válidos**
Denominación	Nombre del local	Campo vacío o local existente.
Puerta de Enlace	Puerta de enlace	Campo vacío, puerta de enlace incorrecta o existente.

Capacidad	Número entero.	Campo vacío o cadena de caracteres.
Tipo de Local	Seleccionar Tipo de local.	Campo vacío.

Caso de uso	**Gestionar Local "Editar local".**
Caso de prueba	Permitir editar un local en la base de datos introduciendo correctamente los datos.
Entrada	El usuario selecciona el local que desea editar e introduce los nuevos datos: Denominación: Lab_303 Puerta de Enlace: 10.31.18.255 Capacidad: 31 Tipo de Local: Laboratorios.
Resultado	El sistema modifica los datos del local en la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar Local "Editar local".**
Caso de prueba	Editar un local introduciendo incorrectamente los datos o dejando algún campo vacío.
Entrada	Existen campos vacíos o con datos incorrectos.
Resultado	El sistema no le permite al usuario entrar datos incorrectos en cada uno de los campos y muestra mensajes de error indicando que los mismos son obligatorios.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Caso de uso	**Gestionar Local "Adicionar local".**
Caso de prueba	Editar un local introduciendo una denominación o puerta de enlace de un local ya existente en la base de datos.
Entrada	El usuario introduce una denominación o puerta de enlace de un local ya existente en la base de datos.
Resultado	El sistema muestra un mensaje de error indicando que ya existe un local con igual denominación y/o puerta de enlace.

Condiciones	Debe existir una denominación o una puerta de enlace con igual nombre en la base de datos.

Prueba del caso de uso Gestionar Tipo de Local escenario "Adicionar Tipo de local".

Condición de entrada	Casos válidos	Casos no válidos
Denominación	Nombre del Tipo de local.	Campo vacío o tipo local existente.
Abreviatura	Cadena de caracteres.	Campo vacío o números.

Caso de uso	**Gestionar Tipo de Local "Adicionar tipo de Local".**
Caso de prueba	Permitir adicionar un tipo de local en la base de datos introduciendo correctamente los datos.
Entrada	El usuario introduce los datos del tipo de local correctamente de la forma: Denominación: Laboratorios. Abreviatura: Lab.
Resultado	El sistema introduce el tipo de local en la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar Tipo de Local "Adicionar tipo de Local".**
Caso de prueba	Adicionar un tipo de local introduciendo incorrectamente los datos o dejando algún campo vacío.
Entrada	Existen campos vacíos o con datos incorrectos.
Resultado	El sistema no le permite al usuario entrar datos incorrectos en cada uno de los campos y muestra mensajes de error indicando que los mismos son obligatorios.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Caso de uso	**Gestionar Tipo de Local "Adicionar tipo de Local".**
Caso de prueba	Adicionar un tipo de local introduciendo una denominación ya existente en la base de datos.
Entrada	El usuario introduce una denominación ya existente en la base de datos.

Resultado	El sistema muestra un mensaje de error indicando que ya existe un tipo de local con igual denominación.
Condiciones	Debe existir una denominación con igual nombre en la base de datos.

Prueba del caso de uso Gestionar Tipo de Local escenario "Editar Tipo de local".

Condición de entrada	**Casos válidos**	**Casos no válidos**
Denominación	Nombre del Tipo de local.	Campo vacío o tipo local existente.
Abreviatura	Cadena de caracteres.	Campo vacío o números.

Caso de uso	**Gestionar Tipo de Local "Editar tipo de Local".**
Caso de prueba	Permitir editar un tipo de local en la base de datos introduciendo correctamente los datos.
Entrada	El usuario selecciona el local que desea editar e introduce los nuevos datos: Denominación: Oficina. Abreviatura: Ofic.
Resultado	El sistema edita el tipo de local en la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar Tipo de Local "Editar tipo de Local".**
Caso de prueba	Editar un tipo de local introduciendo incorrectamente los datos o dejando algún campo vacío.
Entrada	Existen campos vacíos o con datos incorrectos.
Resultado	El sistema no le permite al usuario entrar datos incorrectos en cada uno de los campos y muestra mensajes de error indicando que los mismos son obligatorios.
Condiciones	Deben existir campos vacíos o con datos incorrectos.
Caso de uso	Gestionar Tipo de Local "Editar tipo de Local".
Caso de prueba	Editar un tipo de local introduciendo una denominación ya existente en la base de datos.
Entrada	El usuario introduce una denominación ya existente en la base de datos.

Resultado	El sistema muestra un mensaje de error indicando que ya existe un tipo de local con igual denominación.
Condiciones	Debe existir una denominación con igual nombre en la base de datos.

Prueba del caso de uso Gestionar Usuario escenario "Adicionar Usuario".

Condición de entrada	Casos válidos	Casos no válidos
Nombre	Nombre del usuario.	Campo vacío o usuario existente.

Caso de uso	**Gestionar Tipo de Local "Adicionar tipo de Local".**
Caso de prueba	Permitir adicionar un usuario en la base de datos introduciendo correctamente los datos.
Entrada	El usuario introduce los datos del usuario correctamente de la forma: Nombre: Editor.
Resultado	El sistema introduce el usuario en la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar Tipo de Local "Adicionar tipo de Local".**
Caso de prueba	Adicionar un usuario introduciendo incorrectamente los datos o dejando algún campo vacío.
Entrada	Existen campos vacíos o con datos incorrectos.
Resultado	El sistema no le permite al usuario entrar datos incorrectos en los campos y muestra mensajes de error indicando que los mismos son obligatorios.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Caso de uso	**Gestionar Tipo de Local "Adicionar tipo de Local".**
Caso de prueba	Adicionar un tipo de local introduciendo un usuario ya existente en la base de datos.
Entrada	El usuario introduce un nombre de usuario ya existente en la base de datos.
Resultado	El sistema muestra un mensaje de error indicando que ya existe un usuario con igual nombre.
Condiciones	Debe existir un usuario con igual nombre en la base de datos.

Prueba del caso de uso Gestionar Usuario escenario "Editar Usuario".

Condición de entrada	Casos válidos	Casos no válidos
Nombre	Nombre del usuario.	Campo vacío o usuario existente.

Caso de uso	**Gestionar Tipo de Local "Editar tipo de Local".**
Caso de prueba	Permitir editar un tipo de local en la base de datos introduciendo correctamente los datos.
Entrada	El usuario selecciona el local que desea editar e introduce los nuevos datos: Nombre: admin.
Resultado	El sistema modifica el usuario en la base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar Tipo de Local "Editar tipo de Local".**
Caso de prueba	Editar un usuario introduciendo incorrectamente los datos o dejando algún campo vacío.
Entrada	Existen campos vacíos o con datos incorrectos.
Resultado	El sistema no le permite al usuario entrar datos incorrectos en cada uno de los campos y muestra mensajes de error indicando que los mismos son obligatorios.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

Caso de uso	**Gestionar Tipo de Local "Editar tipo de Local".**
Caso de prueba	Editar un tipo de local introduciendo un usuario ya existente en la base de datos.
Entrada	El usuario introduce un nombre de usuario ya existente en la base de datos.
Resultado	El sistema muestra un mensaje de error indicando que ya existe un usuario con igual nombre.
Condiciones	Debe existir un usuario con igual nombre en la base de datos.

4.3.3 Pruebas de seguridad

Objetivo:

- Nivel de Seguridad de la Aplicación: Verifica que un actor solo pueda acceder a las funciones y datos que su usuario tiene permitido.
- Nivel de Seguridad del Sistema: Verificar que solo los actores con acceso al sistema y a la aplicación están habilitados para accederla.

Garantiza:

- Que los usuarios están restringidos a funciones específicas o su acceso está limitado únicamente a los datos que están autorizados a acceder.
- Que solo aquellos usuarios autorizados a acceder al sistema son capaces de ejecutar las funciones del mismo.

Técnica:

- Identificar cada tipo de usuario y las funciones y datos a los que se debe autorizar.
- Crear pruebas para cada tipo de usuario y verificar cada permiso, creando transacciones específicas para cada tipo de usuario.
- Modificar tipos de usuarios y volver a ejecutar las pruebas.

Criterio de Completitud:

Para cada tipo de usuario conocido, las funciones y datos apropiados y todas las transacciones funcionan como se esperaba.

4.4 Conclusiones Parciales

En este capítulo se mostró la estructura de la aplicación, mediante los modelos de despliegue y de componentes. Se hizo un análisis de las prueba realizadas a la aplicación para comprobar su correcto funcionamiento como un producto ejecutable, que aunque no aseguran la no existencia de fallos, dan una alta confiabilidad, y un nivel de calidad de la aplicación lista para transitar a una etapa de pruebas de aceptación por parte del cliente.

CONCLUSIONES GENERALES

El surgimiento de sistemas informáticos que simplifiquen el problema de obtener un inventario de hardware y software de una red de computadoras es un paso de avance en las tecnologías de la informática, sin embargo, no abarcan todas las necesidades de las empresas de nuestro país, y las más completas no son desarrolladas sobre tecnología libre. Esto llevó al desarrollo del presente trabajo investigativo. Logrando de esa forma, gestionar la obtención de información ofrecida por el Sistema de Gestión de Recursos de Hardware y Software GRHS, así como la gestión de los usuarios, grupos, permisos, módulos y acciones del sistema y mostrar las computadoras distribuidas por locales en una aplicación web. Lo que da cumplimiento a los principales objetivos planteados al inicio. Para lograr este resultado se utilizaron diferentes herramientas: como lenguaje de programación PHP 5.0 y JavaScript, los marcos de trabajo Symfony, Doctrine y Ext JS, el sistema gestor de base de dato fue PostgreSQL y el servidor web Apache. Para realizar el modelado y describir los procesos realizados se utilizó el lenguaje UML y la herramienta CASE Visual Paradigm.

RECOMENDACIONES

Se recomienda continuar la investigación que se ha venido desarrollando para incorporar nuevas funcionalidades a la aplicación como la generación de reportes. Estos reportes podrían incluir un listado de las estaciones de trabajo que no se encuentran en el dominio, las que tienen instalado como Sistema Operativo Linux y de esta forma se optimizarían al máximo las funciones que realiza el sistema y brindará de manera más eficiente la información a los directivos del centro.

Se considera necesaria la implementación de una nomenclatura para nombrar las computadoras de cada local, que incluya el puesto en que la misma se encuentra ubicada físicamente en el laboratorio, para así lograr añadir una funcionalidad al sistema que le permita organizar dichas estaciones por puestos y no por dirección IP, que es como actualmente está implementado.

Otro de los aspectos que se recomienda analizar es la implantación de una herramienta que realice un autodescubrimiento de la red, con el objetivo de incorporar al mapa de red de cada laboratorio, no solo las estaciones sino los dispositivos de red que en él se encuentren.

REFERENCIAS BIBLIOGRÁFICAS

1. **Avila Ramos, Raúl;.** MiTecnologico. [En línea] [Citado el: 8 de Junio de 2011.] http://www.mitecnologico.com/Main/InventariosConceptoYCaracteristicas.
2. BuenasTareas. [En línea] [Citado el: 8 de Junio de 2011.] http://www.buenastareas.com/ensayos/Componentes-De-Hardware/1753383.html.
3. **Dávila Sánchez, Angie Tatiana; Fernanda Álvarez , Luisa;.** CmapsPublic. [En línea] [Citado el: 8 de Junio de 2011.] http://www.google.com/url?sa=t&source=web&cd=5&ved=0CDUQFjAE&url=http%3A%2F%2Fcmapspublic.ihmc.us%2Frid%3D1H29MNNQ9-BDQHZZ-19K5%2Fjj.doc&rct=j&q=Software.%20Se%20conoce%20como%20software%20al%20equipamiento%20l%C3%B3gico%20o%20soporte%20l%C3%B3gico%20&ei=.
4. Universidad Tecnológica Nacional Facultad Regional Mnedoza. [En línea] [Citado el: 8 de Junio de 2011.] http://www.frm.utn.edu.ar/comunicaciones/redes.html.
5. [En línea] [Citado el: 20 de Abril de 2011.] www.segobit.com.
6. [En línea] [Citado el: 20 de Abril de 2011.] www.netsupportdna.com/es.
7. GLPI. [En línea] [Citado el: 20 de Abril de 2011.] http://www.glpi-project.org/?lang=en.
8. [En línea] [Citado el: 20 de Abril de 2011.] www.ocsinventory-ng.org.
9. GuiaLibre. [En línea] [Citado el: 20 de Abril de 2011.] http://guialivre.governoeletronico.gov.br/cacic/sisp2.
10. **Jacobson, I.; Booch, G.; Rumbaugh, J.** *El Proceso Unificado de Desarrollo de Software.* AddisonWesley : s.n., 2000.
11. **Gálvez, Jorge.** Fundamentos de la metodología RUP. [En línea] [Citado el: 27 de Febrero de 2011.] Enero de 2010.] http://www.scribd.com/doc/13071186/Rup..
12. **Pernas, Dimitriades.** *Introducción Práctica a la Administración de Sistemas en Internet.* UNIVERSIDAD DE VALLADOLID : s.n., 1998.
13. **Angel Alvarez, Miguel.** Desarrollo Web. [En línea] [Citado el: 1 de Marzo de 2011.] http://www.desarrolloweb.com/articulos/php-designer.html.
14. **de la Torre, Aníbal.** Adelat. [En línea] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.
15. **González, Rubén Marcos.** Recursos para la programación en php. [En línea] [Citado el: 1 de Marzo de 2011.] http://tgp0607.awardspace.com/Recursos_PHP.pdf..
16. **Denis González, Camilo; Castillo Ruíz, Alié.** *Trabajo de Diploma. Plataforma de Gestión de Servicios Telemáticos en GNU/Linux. Sistema de Inventario de Hardware y Software. Módulo Obtención de Información.* Ciudad de la Habana : s.n., 2010.
17. **Potencier, Fabien; Zaninotto, Francóis;.** *Symfony la guía definitiva.* 2008.
18. Sencha. [En línea] Sencha Inc. [Citado el: 23 de Mayo de 2011.] http://www.sencha.com/.
19. Doctrine. [En línea] [Citado el: 23 de Mayo de 2011.] http://www.doctrine-project.org/.
20. **Kendall, Kenneth E.; Kendall, Julie E.** *Análisis y Diseño del Sistema.* New Jersey : Pearson Education, 2005.
21. **Larman, Craig.** *UML y patrones Tomo II.* s.l. : PEARSO, 2003.
22. Entorno Virtual del Aprendizaje. [En línea] 19 de Abril de 2010. [Citado el: 31 de Mayo de 2011.] http://eva.uci.cu/mod/resource/view.php?id=14103.
23. **Sánchez Mendoza, Maria A.** *Metodologías de Desarrollo de Software.* 2004.
24. UCI. [En línea] UCI. [Citado el: 3 de junio de 2011.] http://www.uci.cu/?q=node/46.