UNIVERSIDAD DE SANCTI SPÍRITUS "JOSÉ MARTÍ PÉREZ"

FACULTAD DE INGENIERÍA

CARRERA INGENIERÍA INFORMÁTICA

SOFTWARE PARA FACILITAR LA GESTIÓN DE INVENTARIOS DE HARDWARE DE LA UNIVERSIDAD DE SANCTI SPÍRITUS JOSÉ MARTÍ PÉREZ

AUTOR: ALIET EXPÓSITO GARCÍA

TUTOR: MSC. ING. BERNARDO LEÓN ÁVILA

SANCTI SPÍRITUS

2014

“Vivimos en una sociedad profundamente dependiente de la ciencia y la tecnología en la que escasamente alguien sabe algo de estos temas. Ello constituye una fórmula segura para el desastre.”

Carl Sagan

Dedicatoria

A mis padres.

No existe obra en el mundo que pudiera dedicarles y refleje todo lo que merecen.

Agradecimientos

Mi vida se la debo a mis padres, literalmente; todo lo que debo agradecerles viene desde el comienzo, hasta hoy. Tengo todas las razones posibles por las cuales darles gracias, pero ellos no hacen todo lo que hacen por el mérito, lo hacen porque son los mejores padres del mundo.

Resumen

Con el objetivo de facilitar la gestión de inventarios de hardware de la Universidad de Sancti Spíritus José Martí Pérez se desarrolla la presente tesis. La ineficiencia e inseguridad del proceso actual hace propicio el nacimiento de este proyecto, que puede contribuir a mejorar considerablemente en confiabilidad y eficiencia el método de hacer el inventario actualmente a las computadoras del centro. En general se propone optimizar el proceso y proveer una plataforma que permita mejoras futuras así como su continuo uso y expansión.

Como metodología de desarrollo se utilizó RUP (*Rational Unified Process*) y UML *(Unified Modeling Language*) como lenguaje de modelado; la arquitectura cliente/servidor, los lenguajes Java, JavaScript (JScript) para escribir el software cliente de extracción de información del hardware; PHP, HTML, SQL como lenguajes para escribir la aplicación web en el servidor. Para la persistencia de datos se escogió el sistema de gestión de base de datos MySQL, pero con posibilidad de migrar a otro SGBD gracias al módulo ‘dbx’ de PHP.

Abstract

The goal of the thesis here presented is to facilitate the hardware inventory management at Universidad de Sancti Spíritus José Martí Pérez. The current methods of making and keeping such inventories is inefficient and insecure, this project was born as a potential solution for transforming the hardware inventory into a reliable and efficient process. This project aims to provide a platform on which the hardware inventory process is expanded and improved through continued use, and ultimately, optimized.

The software development methodology used is RUP (Rational Unified Process) and UML (Unified Modeling Language) is used as a modeling language; a client/server model is also used for the architecture of the software. The programming languages used are Java and JavaScript (in the form of JScript dialect) for the software client which is used in hardware information extraction; PHP, HTML, SQL for the web application running on the server. The database management system selected is MySQL, and PHP's 'dbx' module is used as an abstraction layer between the web application and the DBMS, giving the system the capability of using another DBMS if needed.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Cada institución tiene una serie de recursos que considera “activos fijos”. En términos generales es aquel activo que no está destinado para ser comercializado, sino para ser utilizado, para ser explotado por la empresa o institución. Es un bien que las organizaciones han construido o adquirido con el objetivo de conservarlo para utilizarlo, explotarlo y para ponerlo a su servicio.

Por ende, los equipos de cómputo son considerados como patrimonio de una empresa o entidad gubernamental, estos deben regirse por las políticas de contabilidad y conservación de las mismas. Esto se hace fijando sus ciclos de explotación, los responsables de los medios y la forma en que se controlan los inventarios.

El problema fundamental es que los equipos de cómputo están compuestos por piezas intercambiables a las que en conjunto conocemos por *“hardware”*, y pueden tener elementos externos, como monitor, mouse y teclado (periféricos) o elementos internos como la *motherboard* o placa madre, la memoria RAM, discos duros, etc. Un cambio en uno de estos componentes podría suponer un cambio dramático en las prestaciones de un PC (del inglés *Personal Computer*) y sin embargo este podría lucir exactamente igual por fuera. Por ello cada institución tiene sus propias reglas para evitar hechos delictivos, como sustracciones o alteraciones de los mismos.

En Cuba el Estado pone numerosos medios de cómputo al alcance de sus ciudadanos a través de sus instituciones haciendo grandes sacrificios. El embargo comercial, económico, y financiero impuesto por los Estados Unidos contra Cuba (conocido por nosotros como Bloqueo) pone un freno muy grande a la importación de medios y tecnologías al país, lo que unido a la situación económica por la que atravesamos hace que el cuidado y mantenimiento de todos los medios sea de vital importancia.

La Universidad de Sancti Spíritus “José Martí Pérez” (Uniss) al igual que en muchas instituciones del país, cuenta con una red de computadoras al servicio de los estudiantes y profesores que contribuye a alcanzar los objetivos de cada una de las facultades. Sin embargo esta red ha tenido un crecimiento muy desordenado, lo cual la hace compleja, además podemos encontrar equipos de disímiles fabricantes y generaciones, ubicadas en los edificios de la universidad, que se encuentran distantes físicamente unos de otros.

Algo que se debe tomar en cuenta es que la mayoría de estos ordenadores están conectados a la red y pertenecen al dominio de la Uniss, lo cual facilita el acceso pleno a cada PC desde cualquier punto de la red por los administradores de dominio. Estos tienen privilegios suficientes

como para poder ejecutar aplicaciones que muestren todo el hardware de cada PC, se podría pensar en una aplicación que con estos privilegios automatizara el proceso.

La Oficina de Seguridad para las Redes Informáticas (OSRI) es el órgano encargado de llevar a cabo y velar por lo descrito en la resolución No. 127 /2007 del antiguo MIC (Ministerio de la Informática y las Comunicaciones) la cual establece el reglamento para la seguridad informática de las TIC (Tecnologías de la Informática y las Comunicaciones).

El incremento cada vez mayor de las tecnologías de la información en las disímiles esferas de la sociedad a nivel mundial, muchas labores antes realizadas manualmente por el hombre se han sustituido por sistemas informáticos con el objetivo de disminuir el error humano, permitir el almacenamiento de una gran cantidad de información, agilizar determinados procesos y permitir un mayor y mejor alcance de la información. Por lo que las tecnologías de la información se han convertido en un medio imprescindible para lograr la eficiencia de la gestión de la información en cualquier institución.

En la Uniss, la auditoría de los equipos informáticos es una necesidad para mantener la integridad de los medios materiales y la seguridad informática, por lo que este reglamento establece que cada PC debe estar acompañada de una “ficha técnica” (Anexo 1), que describe sus características enumerando cada componente de la misma, así como una lista del software que tiene instalado. Cada área de la Uniss cuenta con un responsable de seguridad informática que es el encargado de llevar el control de inventarios de todos los componentes internos de los equipos de cómputos mediante dicho documento.

Un paso más en el control es que cada ordenador tiene puesto un sello en el chasis con la firma del responsable, que si es roto sin aviso puede ser señal de una violación. No obstante, si un cambio de hardware es realizado como motivo de robo u otro evento no autorizado, este no puede ser detectado hasta que se concilie con el representante de seguridad informática del área o de la Uniss, lo cual puede hacerse en intervalos de hasta meses. Además, aunque lo que esté regulado que cada PC esté acompañado por su ficha técnica, en la práctica no siempre se hace así, pues cada responsable de PC no siempre lleva o actualiza el documento, que está siempre expuesto a pérdida o alteración, poniendo en riesgo la integridad del hardware.

Además, la visión de la Uniss tiene 2 puntos que nos hacen ver cuán grande sería el impacto de un sistema que contribuyera a la gestión de inventarios de hardware:

- La infraestructura responde al desarrollo de los procesos sustantivos de la Uniss con acciones concretas y con una red que asegura el empleo de las TIC.
- La prevención y el control hacia cualquier tipo de manifestación de corrupción, ilegalidad, fraude, delito o vicio, forman parte de la cultura organizacional. El sistema de control interno se consolida.

O sea, se prevé un creciente desarrollo del número de equipos de cómputo y puntos de acceso a la red, lo que implica que la protección de dichos medios, así como la lucha contra la corrupción e ilegalidades es un ambiente cada vez más complejo.

Por lo que se nos plantea la siguiente **situación problemática**: En la Uniss el control de los inventarios de los medios informáticos se hace de manera ineficiente, inconsistente e insegura, al hacerse de forma manual. Muchas veces estos inventarios pueden estar desactualizados y pueden comprometerse, extraviarse o dañarse. Percatarse de un cambio de hardware en una PC puede tardar mucho tiempo y actualizar toda la Uniss es un trabajo engorroso.

Esto nos lleva al siguiente **problema de investigación**: ¿Cómo desarrollar un software que facilite la gestión de inventarios de hardware de la Uniss?

Como **objetivo general**: Desarrollar un software que facilite la gestión de inventarios de hardware de la Uniss.

Para alcanzar el objetivo general propuesto, dirigir la investigación e intentar dar solución al problema de investigación se formularon las siguientes **preguntas de investigación**:

1. ¿Qué fundamentos teóricos y metodológicos sustentan la elaboración de un software que facilite la gestión de inventarios de hardware?
2. ¿Cómo diseñar un software que facilite la gestión de inventarios de hardware en la Uniss?
3. ¿Cómo implementar un software que facilite la gestión de inventarios de hardware en la Uniss?

Para dar cumplimiento a los objetivos, se desarrollaron las siguientes **tareas de investigación**:

1. Determinar los fundamentos teóricos y metodológicos que sustentan la elaboración de un software que facilite la gestión de inventarios de hardware.
2. Diseñar un software que facilite la gestión de inventarios de hardware en la Uniss.
3. Implementar un software que facilite la gestión de inventarios de hardware en la Uniss.

El trabajo posee una introducción, tres capítulos, conclusiones, recomendaciones, bibliografías y anexos.

Capítulo 1. Fundamentación teórica y metodológica que sustenta el desarrollo de un software para facilitar la gestión de inventarios de hardware en la Universidad de Sancti Spíritus José Martí Pérez.

En este capítulo abordaremos los aspectos teóricos relacionados con el tema a desarrollar, exponiendo los principales conceptos asociados al dominio del mismo. Se describe el contexto donde se enmarca, las características y dificultades que lo acompañan, así como las metodologías y tecnologías utilizadas.

Capítulo 2. Descripción del software propuesto para la gestión de inventarios de hardware en la Universidad de Sancti Spíritus José Martí Pérez.

Tomando como guía la Metodología RUP, en el segundo capítulo se utilizan algunos de los artefactos disponibles: el modelo de negocio, requerimientos funcionales y no funcionales, los diagramas de casos de uso y la descripción de cada uno, los cuales ayudan a modelar y describir la solución propuesta. Además, se presenta una descripción detallada de las reglas de negocio con el objetivo de asegurar el cumplimiento de las restricciones que existen en el negocio.

Capítulo 3. Construcción de la aplicación propuesta para la gestión de inventarios de hardware en la Universidad de Sancti Spíritus José Martí Pérez.

En el tercer capítulo se muestran los resultados de la etapa de implementación del sistema. Se desarrollan los diagramas de clases, el diseño de la base de datos, el diagrama de despliegue y el de componentes. Se describen los principios de diseño seguidos para la interfaz de usuario, la seguridad del sistema, el tratamiento de excepciones, la concepción de la ayuda y los estándares de codificación.

Capítulo 1. Fundamentación teórica y metodológica que sustenta el desarrollo de un software para facilitar la gestión de inventarios de hardware en la Universidad de Sancti Spíritus José Martí Pérez.

Introducción

El uso de las tecnologías de la información y las comunicaciones es primordial para la expansión y supervivencia de cualquier organización, pues constituye un elemento imprescindible que posibilita un continuo desarrollo; la Uniss lograría fortalecer el control de inventarios de sus medios informáticos con el desarrollo de un software que facilite la gestión de inventarios de hardware. Es por ello que el presente capítulo contiene la fundamentación teórica sobre el tema a desarrollar, se realiza un estudio sobre las tecnologías, lenguajes y herramientas existentes, determinando cuáles van a ser las utilizadas en el desarrollo del software lo que posibilitará enfrentar exitosamente la situación problemática.

1.1 La gestión de la información

La información está constituida por un grupo de datos ya supervisados y ordenados, que sirven para construir un mensaje basado en un cierto fenómeno o ente. La misma permite resolver problemas y tomar decisiones, ya que su aprovechamiento racional es la base del conocimiento. (Marcial, 1996)

La evolución en el uso de la información ha demostrado que es imprescindible en la vida moderna, pero también lo es su correcto proceso, por lo que su adecuada gestión adquiere un valor decisivo.

Se entiende por gestión: el proceso mediante el cual se obtiene, despliega o utiliza una variedad de recursos básicos para apoyar los objetivos de la organización. (Dante, 1998)

La finalidad de la gestión de la información es ofrecer mecanismos que permitan adquirir, producir y transmitir, al menor coste posible, datos e informaciones con una calidad, exactitud y actualidad suficientes. En términos perfectamente entendibles sería conseguir la información adecuada, para la persona que lo necesita, en el momento que lo necesita, al mejor precio posible para tomar la mejor de las decisiones. (Morales Flores, 2007)

Según otro autor la gestión de la información es un proceso que incluye operaciones de extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes y que gestiona el acceso y los derechos de los usuarios sobre el sistema. (Curto, 2006)

Para desarrollar una correcta gestión de la información es necesario tener en cuenta una serie de pasos, entre los que se encuentran los siguientes: (Bartle, 2009)

- Determinar la información que se precisa.
- Recoger y analizar la información.
- Registrarla y recuperarla cuando sea necesaria.
- Utilizarla.
- Divulgarla.

La llegada de las tecnologías de la información y las comunicaciones (TIC) y su incalculable avance ha facilitado la gestión de la información y se encuentran estrechamente relacionadas; TIC es un término que contempla toda forma de tecnología usada para crear, almacenar, intercambiar y procesar información en sus varias formas, tales como datos, conversaciones de voz, imágenes fijas o en movimiento, presentaciones multimedia y otras formas, incluyendo aquéllas aún no concebidas. En particular, las TIC están íntimamente relacionadas con computadoras, software y telecomunicaciones. Su objetivo principal es la mejora y el soporte a los procesos de operación y negocios para incrementar la competitividad y productividad de las personas y organizaciones en el tratamiento de cualquier tipo de información. (Leal, 2008).

Según (Elizalde, 2004) las TIC constituyen el reto del desarrollo de la sociedad de la información y el conocimiento en todo el mundo a través de programas que tienen como objetivo ofrecer servicios y aplicaciones que permitan el acceso de las personas desde donde estén a los lugares donde se encuentran aquellas informaciones que les interesan, y que puedan obtenerlas con el menor esfuerzo por su parte.

1.2 Inventario

El *inventario* constituye una reserva de materiales, materias primas, producción en procesos o productos terminados, que no tiene un empleo sistemático y son originados por la baja fiabilidad,

para garantizar un determinado servicio al cliente. (Cespón, 2000).

Es la recopilación de todos los bienes y materiales que posee la universidad, a fin de comparar las existencias reales con los registros contables.

Tipos de inventarios

Inventario del hardware

Inventario del software

Inventario de consumibles

Inventario de documentos

Inventario de inmuebles, instalaciones, mobiliario y equipos de sistemas

Inventario del personal

Inventario de bases de datos e información institucional

Inventario del hardware

El propósito es evaluar su uso adecuado, su resguardo, su aprovechamiento y el estado en que éstos se encuentran; incluyendo las instalaciones internas y los componentes físicos de los sistemas computacionales.

El inventario del hardware se realiza en dos partes:

Inventario físico del hardware.

Registros existentes del inventario del hardware

Inventario físico del hardware

Es el recuento físico de todos los sistemas computacionales instalados en el área de sistemas, se hace con el propósito de identificar el total de hardware que tiene la universidad.

El hardware debe estar registrado en los documentos formales de la entidad, ya sea los registros contables, facturas o en cualquier otro registro que avale que es propiedad de la universidad.

1.3 Metodologías para el desarrollo de software

Una metodología de desarrollo de software en ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Consiste en una filosofía de desarrollo de programas de computación con el enfoque del proceso de desarrollo de software, así como en las herramientas, modelos y métodos para asistir dicho proceso. (CMS Information Technology, 2008)

1.3.1 Metodologías ágiles y tradicionales

La calidad en el desarrollo de un sistema informático es imprescindible, para ello es necesario seguir las indicaciones de alguna metodología. Antes de llevar a cabo el proceso de desarrollo, se debe hacer un estudio de las tecnologías existentes en la actualidad, conocidas o no, con el fin de utilizar la más conveniente para el problema existente.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software, tienen como objetivo presentar un conjunto de técnicas tradicionales y modernas de modelado de sistemas que permitan desarrollar software de calidad, incluyendo heurísticas de construcción y criterios de comparación de modelos de sistemas.

Las metodologías tradicionales nos ofrecen una buena solución para proyectos donde el entorno es estable y donde los requisitos se conocen con exactitud y están bien definidos, porque no están pensadas para trabajar con incertidumbre. La mayoría de las decisiones se toman al principio.

Las metodologías ágiles tienen sus inicios a principios de la década de 1990, con el surgimiento de un enfoque que fue bastante revolucionario para su momento ya que iba en contra de toda creencia de que mediante procesos altamente definidos se iba a lograr obtener software en tiempo, costo y con la requerida calidad. En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace formalmente el término “ágil” aplicado al desarrollo de software. Están pensadas para proyectos con pocos principios, requisitos cambiantes, donde no exista un contrato formal, o el mismo sea flexible. (Calderón, 2009)

Ambas metodologías tienen ventajas y desventajas y no deben ser comparadas sino determinar cuál es la más adecuada para nuestro proyecto. Según los conceptos antes analizados y las características del negocio en cuestión se determina que la metodología tradicional es la

apropiada ya que este proyecto tiene numerosos requisitos que están bien definidos y el cliente no es parte del equipo de desarrollo.

1.3.2 Metodología *Rational Unified Process* (RUP)

Se analiza emplear la metodología RUP porque se adapta a las características del negocio que se lleva a cabo en la Uniss además porque brinda facilidades para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. Además es una metodología con la capacidad de adaptación a cualquier proyecto, cuyo objetivo es producir software de alta calidad, cumplimentando los requerimientos de los usuarios dentro de una planificación y presupuesto establecidos. (Rumbaugh, Booch, & Jacobson, 2006)

En otras palabras, es un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Es un marco de trabajo genérico que puede especializarse, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. Fue creado por un grupo de estudiosos de la Ingeniería de Software formado por Ivar Jacobson, Grady Booch y James Rumbaugh en el año 1998. Es un proceso basado en componentes y utiliza UML para preparar todos los esquemas de un sistema software. No obstante, los verdaderos aspectos definitorios de RUP se resumen en tres frases clave: está dirigido por casos de uso, centrado en la arquitectura y es iterativo e incremental. Además cubre el ciclo de vida de un proyecto y toma en cuenta las mejores prácticas a utilizar en el modelo de desarrollo de software. (Rumbaugh, Booch, & Jacobson, 2006)

1.3.3 Lenguaje de Modelado

Los lenguajes de modelado son usados en combinación con las metodologías de desarrollo de software para avanzar de una descripción inicial a un plan de implementación y comunicar dicho plan a los desarrolladores. Con el uso de lenguajes de modelado el ingeniero de software va teniendo una visión del sistema a construir por lo que es de gran ayuda hacer uso de los mismos. (Hernández, 2010).

Lenguaje Unificado de Modelado UML

El Lenguaje Unificado de Modelado (*Unified Modeling Language*, UML por sus siglas en inglés). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. UML es

un lenguaje de modelado para especificar o para describir métodos o procesos. Se utiliza para definir un sistema y detallar los artefactos en el mismo. (Manage, 2007).

1.4 Tendencias actuales empleadas en el diseño del software propuesto.

Las tendencias y tecnologías actuales vienen dadas por el desarrollo de la informática en nuestro país, estas tendencias son globales, producto del impacto de nuestros científicos y del constante perfeccionamiento y alto nivel de nuestros ingenieros. La elección de software de licencias libres, de paradigmas y patrones de arquitectura que son novedosos en el mundo, evidencian el alto nivel y el rigor de los profesionales en formación de nuestro centro.

1.4.1. Programación Orientada a Objetos

La Programación Orientada a Objetos (POO) es un paradigma en el sentido de que se trata de "(…) un conjunto de teorías, estándares y métodos que juntos representan una forma de organizar el conocimiento, esto es, una forma de ver el mundo." (Introducción a la Programación Orientada a Objetos, 2004)

Los conceptos de POO se remontan a los orígenes de Simula 67, lenguaje diseñado por Ole-Johan Dhal y Krysten Nygaard en la Universidad de Oslo y el Centro de Computación Noruego en 1967. Este lenguaje, si bien no alcanzó su gloria durante años, a principios de la década del 70 con la aparición de Smalltalk y C++, en 1986, que se les reconoció el mérito de sus creadores (Morero, 2000). Su eje central se encuentra en los objetos, como resultado de la simulación de la realidad objetiva, los cuales poseen **responsabilidades** y se comunican con los demás objetos a través de **mensajes**. Un objeto es una **encapsulación** del estado (valores de datos) y del **comportamiento** (operaciones) determinado por la **clase** a la cual pertenece el objeto. Dicho comportamiento se exhibirá mediante la invocación de un **método** como respuesta a un mensaje.

La POO introduce el principio de **herencia**, donde:

Las clases se pueden organizar en una estructura de herencia jerárquica. Una subclase heredará atributos de una superclase que esté más arriba en el árbol. Una superclase abstracta es una clase (…) que se usa sólo para crear subclases y para la cual no hay ejemplares [objetos] directos. (Introducción a la Programación Orientada a Objetos, 2004*)*

Este principio permite que diferentes tipos de datos compartan el mismo código, el cual puede ser adaptado para que se ajuste a las circunstancias específicas de cada tipo de datos individuales mediante el mecanismo de **polimorfismo**.

Al reducir la interdependencia entre los componentes de software (**módulos**), la POO permite el desarrollo de sistemas de software reutilizable. Tales componentes pueden crearse y probarse como unidades independientes, aisladas de otras partes de una aplicación de software (**principio de ocultación**). Además, permiten al programador lidiar con problemas en un nivel más alto de **abstracción**, definir y manipular objetos simplemente en términos de los mensajes que los componentes entienden y de una descripción de las tareas que ellos realizan, pasando por alto los detalles de la implantación (Introducción a la Programación Orientada a Objetos, 2004).

Hoy en día, casi todos los lenguajes más usados en los 80 se han reconvertido en mayor o menor medida a la POO. Tal es el caso de C, que ha pasado a llamarse C++, quien constituye uno de las mejores implementaciones de la POO sobre un lenguaje compilado. Su punto más conflictivo es que permite herencia múltiple, lo que, según los teóricos, no es muy aconsejable. Pascal ha sido reciclado por Borland, pasando a llamarse Delphi, el cual tiene una implementación de la POO aceptable, pero al no permitir la sobrecarga de funciones pierde una de las características de la POO: el polimorfismo. Basic ha pasado a llamarse Visual Basic en manos de Microsoft, este, aun siendo uno de los lenguajes más famosos del momento, no es un lenguaje de la POO completo, ya que no incluye la característica más importante de la POO: la herencia. Java es la estrella de los últimos años: es pura POO, impecablemente concebido e implementado por Sun. Incluso el viejo Cobol se ha reciclado, existiendo en estos momentos más de una versión de Cobol que incluye la POO (Morero, 2000).

1.4.2. Software libre y de código abierto

El **software libre y de código abierto** (también conocido como ***FOSS*** o ***FLOSS***, siglas de *free/libre and open source software*, en inglés) es el software que está licenciado de tal manera que los usuarios pueden estudiar, modificar y mejorar su diseño mediante la disponibilidad de su código fuente. (The OSI Approved Licenses)

El software libre y el software de código abierto si bien comparten modelos de desarrollo similares, tienen diferencias en sus aspectos filosóficos. El software libre se enfoca en las libertades filosóficas que les otorga a los usuarios mientras que el software de código abierto se enfoca en

las ventajas de su modelo de desarrollo. (Software libre para una sociedad libre) "FOSS" es un término imparcial respecto a ambas filosofías.

El término de “Software Libre” aparece por primera vez a principios de la década del 80 por Richard Stallman, creador de la *Free Software Foundation* (FSF) en 1985. Este término “(…) significa que el software respeta la libertad de los usuarios y la comunidad. En términos generales, los usuarios tienen la libertad de copiar, distribuir, estudiar, modificar y mejorar el software. Con estas libertades, los usuarios (tanto individualmente como en forma colectiva) controlan el programa y lo que hace.” (Arteaga Mejía, 2012).

En software libre, la palabra "libre" en el nombre no se refiere al precio; se refiere a la libertad. Primero, a la libertad de copiar y redistribuir un programa a tus vecinos, para que ellos al igual que tú, lo puedan usar también. Segundo, a la libertad de cambiar un programa, así podrás controlarlo en lugar que el programa te controle a ti; para esto, el código fuente tiene que estar disponible para ti. (Free Software Foundation, 1986)

Las libertades que definen al software libre, son:

1. La libertad de ejecutar el programa, para cualquier propósito (**libertad 0**).
2. La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera (**libertad 1**).
3. La libertad de redistribuir copias para que pueda ayudar al prójimo (**libertad 2**).
4. La libertad de mejorar el programa y publicar sus mejoras, y versiones modificadas en general, para que se beneficie toda la comunidad (**libertad 3**).

Es necesario enfatizar que "el acceso al código fuente es una condición necesaria en las libertades 1 y 3. (La Definición de Software Libre)

El **software de código abierto** (en inglés *open source software* u **OSS**) es el software cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de software compatible con la Open Source Definition o forman parte del dominio público. Esto permite a los usuarios utilizar, cambiar, mejorar el software y redistribuirlo, ya sea en su forma modificada o en su forma original. Frecuentemente se desarrolla de manera colaborativa y los resultados se publican en internet. (La definición de Open Source)

“Mucha gente utiliza la expresión software de «código abierto» para referirse, más o menos, a la misma categoría a la que pertenece el software libre. Sin embargo, no son exactamente el mismo

tipo de software: ellos aceptan algunas licencias que nosotros consideramos demasiado restrictivas, y hay licencias de software libre que ellos no han aceptado. Sin embargo, las diferencias entre lo que abarcan ambas categorías son pocas: casi todo el software libre es de código abierto, y casi todo el software de código abierto es libre. Nosotros preferimos la expresión «software libre» porque se refiere a libertad, algo que la expresión «código abierto» no hace." (Free Software Foundation)

Para el proyecto aquí propuesto se seleccionó la Licencia Pública General de GNU, lo que lo convierte en software libre y de código abierto. Se desea así aportar a la comunidad un granito de arena, retroalimentando y agradeciendo por las facilidades y funcionalidades de todo el software que se escogió para su desarrollo.

1.4.3. Modelo-Vista-Controlador

Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software descrito por primera vez en 1979 por Trygve Reenskaug, que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos: la vista, el controlador y el modelo. La vista maneja la salida gráfica y/o textual a la porción de la pantalla asignada a la aplicación. El controlador interpreta las entradas del usuario por el ratón o el teclado e invoca peticiones al modelo y/o a la vista, según sea apropiado. Finalmente, el modelo administra el comportamiento y los datos del dominio de la aplicación, responde a las peticiones de información acerca de su estado (por lo general desde la vista) y responde a las instrucciones para cambiar el estado (por lo general desde el controlador) (Burbeck, 1992).

El MVC 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 interfaz 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.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista), logrando simplificar el mantenimiento de las aplicaciones. En ella, el controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). Por otra parte, el modelo se encarga de la abstracción de la lógica

relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación (Potencier y Zaninotto, 2008).

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1. El usuario realiza una acción en la interfaz.

2. El controlador trata el evento de entrada (previamente se ha registrado).

3. El controlador notifica al modelo la acción del usuario, lo que puede implicar un cambio del estado del modelo (si no es una mera consulta).

4. Se genera una nueva vista. La vista toma los datos del modelo (el modelo no tiene conocimiento directo de la vista).

5. La interfaz de usuario espera otra interacción del usuario, que comenzará otro nuevo ciclo.

1.5 Modelo cliente/servidor

El modelo cliente/servidor (C/S) es una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realiza se efectúa con la mayor eficiencia posible y permita simplificar las actualizaciones y mantenimiento del sistema.

Es un modelo distribuido donde el software está dividido entre:

- Tareas servidor.
- Tareas cliente.

Cuando se utiliza un servicio en red, como consultar una base de datos, transferir un fichero o participar en un foro de discusión, se establece un proceso en el que entran en juego dos partes. Por un lado, el usuario, quien ejecuta una aplicación en el ordenador local: el denominado programa cliente. Este programa cliente se encarga de ponerse en contacto con el ordenador remoto para solicitar el servicio deseado. El ordenador remoto por su parte responderá a lo solicitado mediante un programa que está ejecutando. Este último se denomina programa servidor. Los términos cliente y servidor se utilizan tanto para referirse a los programas que cumplen estas funciones, como a los ordenadores donde son ejecutados esos programas.

El cliente es el que inicia un requerimiento de servicio, el remitente de una solicitud. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de la red. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente. Es quien inicia solicitudes o peticiones, tiene por tanto un papel activo en la comunicación. Espera y recibe las respuestas del servidor. Por lo general, puede conectarse a varios servidores a la vez. Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

El servidor es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente, es el receptor de la solicitud enviada por el cliente. Los servidores pueden estar conectados a los clientes a través de redes, para proveer de múltiples servicios a los clientes y usuarios tales como impresión, acceso a bases de datos, correo, aplicaciones web, procesamiento de imágenes, multimedia, etc.

El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.

Las funciones que lleva a cabo el servidor se resumen en los siguientes puntos:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

El programa o los programas cliente que el usuario utiliza para acceder a los servicios de Internet realizan dos funciones distintas. Por una parte, se encargan de gestionar la comunicación con el ordenador servidor, de solicitar un servicio concreto y de recibir los datos enviados por éste; y por otra, es la herramienta que presenta al usuario los datos en pantalla y que le ofrece los comandos necesarios para utilizar las prestaciones que ofrece el servidor.

Los demandantes (clientes) y proveedor (servidor) cumplen con un papel que es parte del proceso de funcionamiento del software:

Papel del cliente:

- Inicia el diálogo.
- Envía peticiones al servidor conforme a algún protocolo asimétrico.
- Pide que el servidor actúe, o que le informe, o ambas cosas.

Papel del servidor:

- Espera pasivamente peticiones de los clientes.
- Responde a las peticiones según las políticas definidas y provee las funcionalidades para las cuales fue programado.

La aplicación propuesta usa un modelo C/S, donde por una parte tenemos el software cliente que hace el inventario de hardware a cada computadora y que a su vez envía la información encuestada hacia el servidor donde es almacenada en la base de datos; y por otra parte tenemos el cliente que usará el encargado de gestionar la información, el jefe de seguridad informática, en la forma de una computadora con navegador web que utilizará para conectarse al servidor y realizar sus tareas.

Aplicaciones web

Con el avance de la tecnología y las redes de comunicación las aplicaciones web emergieron rápidamente hacia una posición dominante; son aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. (Palacios, 2008)

Tanto las capacidades de las tecnologías cliente, como las tecnologías de servidor, para procesar una página Web, han sido enriquecidas, optimizadas, estandarizadas, en definitiva hoy en día son más potentes, dinámicas e integrables. (Palacios, 2008)

1.6 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que puedan ser interpretados y ejecutados por ordenadores. Este tipo de lenguajes pueden usarse para crear sistemas software que controlen el comportamiento físico y lógico de una computadora, permitan calcular algoritmos complejos con alta precisión, o que faciliten la comunicación entre las personas.

Actualmente existen diversos lenguajes de programación que permiten el desarrollo de aplicaciones con tecnologías web. Estos se pueden clasificar como lenguajes del lado del cliente y del lado del servidor. En este epígrafe se presentarán los que se usarán en el desarrollo de la propuesta de solución.

1.6.1 Lenguajes del lado del cliente

Las tecnologías del lado del cliente están orientadas preferentemente, como su nombre indica, para ejecutarse en las computadoras cliente, por ejemplo Java y JavaScript, entre otros. Esto proporciona las capacidades del cliente que hacen posible la creación de aplicaciones dinámicas de Internet al aprovechar el poder de procesamiento local de las computadoras y los dispositivos, además del acceso a la información de hardware requerida para el inventario.

Para computadoras ejecutando el sistema operativo Linux se escogió el lenguaje Java debido a su propiedad de ejecutarse donde quiera que exista un entorno de ejecución de Java. La portabilidad de Linux y su capacidad de instalarse en entornos que difieren extensamente de una distribución a otra, hace difícil tener una única aplicación genérica que se pueda ejecutar en todas las computadoras, Java elimina la necesidad de tener que recompilar el software cliente para diferentes versiones de kernels de Linux, o diferentes arquitecturas de procesador, incluso en distribuciones completamente diferentes. En Linux también existen compiladores de código nativo de Java, como GCJ, lo que permite compilar este código objeto de Java a código binario si se quisiera. Java es además un lenguaje que provee muchas facilidades para la programación de aplicaciones que utilizan la red, una facilidad primordial a la hora de escribir el software cliente.

Para computadoras ejecutando el sistema operativo Microsoft Windows se escogió el lenguaje JavaScript en la forma del dialecto JScript, lo que permite acceder a toda la información de hardware de la computadora utilizando solo el navegador Internet Explorer incluido en todas las versiones de dicho sistema operativo, eliminando la necesidad de compilar el software cliente y asegurándose la portabilidad del software cliente en todas las versiones.

A continuación se mencionan en más detalle las tecnologías del lado del cliente a emplear.

Java

Java es un lenguaje de desarrollo de propósito general, y como tal es válido para realizar todo tipo de aplicaciones profesionales; desarrollado por la empresa Sun Microsystems en 1995. Incluye una combinación de características que lo hacen único y está siendo adoptado por multitud de fabricantes como herramienta básica para el desarrollo de aplicaciones comerciales de gran repercusión. (Belmonte, 2009)

Una de las características más importantes es que los programas “ejecutables”, creados por el compilador de Java, son independientes de la arquitectura. Se ejecutan indistintamente en una gran variedad de equipos con diferentes microprocesadores y sistemas operativos. Permite

escribir applets (pequeños programas que se insertan en una página HTML) y se ejecutan en el ordenador local. Se pueden escribir aplicaciones cliente/servidor, aplicaciones distribuidas en redes locales y en Internet. Es fácil de aprender y está bien documentado. Las aplicaciones son fiables. Puede controlarse su seguridad frente al acceso a recursos del sistema y es capaz de gestionar permisos y criptografía. También, según Sun, la seguridad frente a virus a través de redes locales e Internet está garantizada. Java es un lenguaje de propósito general, puede programarse en él cualquier cosa. Es un lenguaje bien estructurado, sin punteros y sin necesidad de tener que controlar la asignación de memoria a estructuras de datos u objetos. (Belmonte, 2009)

Características del lenguaje.

- Es intrínsecamente orientado a objetos.
- Funciona perfectamente en red.
- Aprovecha características de la mayoría de los lenguajes modernos evitando sus inconvenientes. En particular los del C++.
- Tiene una gran funcionalidad gracias a sus bibliotecas (clases).
- No tiene punteros manejables por el programador, aunque los maneja interna y transparentemente.
- El manejo de la memoria no es un problema, la gestiona el propio lenguaje y no el programador.
- Genera aplicaciones con pocos errores posibles.

El lenguaje Java puede considerarse como una evolución del C++. La sintaxis es parecida a la de este lenguaje. A pesar de que puede considerarse como una evolución del C++ no acarrea los inconvenientes del mismo, ya que Java fue diseñado “partiendo de cero”, es decir, no necesitaba ser compatible con versiones anteriores de ningún lenguaje como ocurre con C++ y C. Gracias a que fue diseñado “partiendo de cero” ha conseguido convertirse en un lenguaje orientado a objetos puro, limpio y práctico. No permite programar mediante otra técnica que no sea la programación orientada a objetos (POO)

Bien es sabido que la mayoría de los errores en las aplicaciones vienen producidos por fallos en la gestión de punteros o la asignación y liberación de memoria y Java soluciona este problema. Además, el lenguaje contiene estructuras para la detección de excepciones (errores de ejecución previstos) y permite obligar al programador a escribir código fiable mediante la declaración de excepciones posibles para una determinada clase reutilizable. Constituye uno de los lenguajes

de programación más utilizados en el desarrollo de aplicaciones de código abierto y posee una variada bibliografía para aquellos programadores interesados en el aprendizaje de este lenguaje. (Belmonte, 2009)

JavaScript y JScript

JavaScript es un lenguaje dinámico de propósito general sensible al contexto, basado en objetos, orientado al documento y a eventos, es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java pero siendo completamente diferente. Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Se puede crear todo tipo de programa que puede ser ejecutado en varios sistemas operativos, como Linux, Microsoft Windows y MacOS X. El mismo permite a los desarrolladores crear acciones en sus páginas web, creando diferentes efectos e interactuando con los usuarios.

Los programas escritos con JavaScript se pueden probar directamente en cualquier navegador, sin necesidad de procesos intermedios (Mateu, 2004).

JavaScript permite crear una interfaz de usuario activa. Además, posibilita controlar el navegador, verificar la presencia de plug-ins que pueden resultar necesarios para determinadas aplicaciones y activar estos componentes en caso de estar inhabilitados (Negrino y Smith, 2000).

Prácticamente todos los sistemas operativos cuentan al menos con un intérprete de JavaScript, su gran popularidad es debida a su uso como el lenguaje interpretado de la web.

JScript es el dialecto de JavaScript que utiliza Microsoft para su navegador Internet Explorer. La diferencia entre JScript y JavaScript es mínima, siendo nombres diferentes para el mismo lenguaje de programación; Microsoft decidió cambiarle el nombre a JScript para evitar problemas de marcas registradas. Es implementado como un motor de *Active Scripting* lo que le permite ejecutarse en aplicaciones que soporten *Active Scripting* como Internet Explorer, *Active Server Pages* y *Windows Scripting Host*. Fue incluido por primera vez en Internet Explorer 3.0 en agosto de 1996 y su versión más reciente es JScript 9.0, incluido en Internet Explorer 9. El acceso a hardware se hace a través de los objetos ActiveX de la plataforma *OLE/COM automation* que permiten su acceso desde la red a la computadora local y es precisamente este acceso a objetos ActiveX una de las diferencias entre los dos lenguajes.

1.6.2 Lenguajes del lado del servidor

Los lenguajes del lado del servidor proporcionan un entorno rápido de creación de scripts y soporte para los estándares más importantes. Además de las aplicaciones tradicionales de bases de datos, las aplicaciones dinámicas de Internet prometen la integración de las comunicaciones bidireccionales y los datos en tiempo real en las aplicaciones, en este sentido, PHP se perfila como un ejemplo necesario del lado del servidor.

PHP

PHP, acrónimo recursivo de *Hypertext Preprocessor*, es un lenguaje sencillo, de sintaxis cómoda y similar a la de otros lenguajes como Perl, C y C++. Es rápido, interpretado, orientado a objetos y multiplataforma. Dispone de una multitud de bibliotecas. Además, es un lenguaje ideal para aprender a desarrollar aplicaciones web. Cuenta con diversos módulos que dotan a los programadores de un arsenal de herramientas libres para la creación de aplicaciones.

Este lenguaje suele ser utilizado conjuntamente con Perl, Apache, MySQL o PostgreSQL, formando una combinación barata, potente y versátil. Apache y otros servidores web, Roxen entre ellos, puede incorporar PHP como un módulo propio del servidor, lo cual permite que las aplicaciones escritas en PHP resulten mucho más rápidas que las aplicaciones CGI habituales. (Mateu, 2004)

Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF e incluso películas Flash. Es un lenguaje extremadamente simple que ofrece facilidades de trabajo a los desarrolladores. Aunque el desarrollo con PHP está centrado en la programación del lado del servidor, se puede utilizar para muchas otras tareas. (Zmievski y otros, 2001)

Por otro lado el código escrito en PHP es independiente de la plataforma, esto quiere decir que una aplicación escrita en este lenguaje puede ser ejecutada en cualquier sistema operativo, dependiendo solo de que el módulo esté presente en el servidor HTTP. Presenta un gran número de funciones predefinidas, es de fácil aprendizaje por su similitud con otros lenguajes de programación y se puede integrar de manera sencilla con múltiples bases de datos.

SQL *(Structured Query Language)*

El SQL es el lenguaje estándar ANSI/ISO de definición, manipulación y control de bases de datos relacionales. Es un lenguaje declarativo: solo hay que indicar qué se quiere hacer. En cambio, en los lenguajes procedimentales es necesario especificar cómo hay que hacer cualquier acción

sobre la base de datos. El SQL es un lenguaje muy parecido al lenguaje natural; concretamente, se parece al inglés, y es muy expresivo. Por estas razones, y como lenguaje estándar, el SQL es un lenguaje con el que se puede acceder a la gran mayoría de los sistemas relacionales.

SQL permite la concesión y denegación de permisos, la implementación de restricciones de integridad y controles de transacción, y la alteración de esquemas. Debido a que es un lenguaje declarativo, especifica qué es lo que se quiere y no como conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución. Posibilita lanzar consultas con el fin de recuperar información de interés de una base de datos de una forma sencilla.

A principio de los años setenta, los laboratorios de investigación Santa Teresa de IBM empezaron a trabajar en el proyecto System R. El objetivo de este proyecto era implementar un prototipo de SGBD relacional; por lo tanto, también necesitaban investigar en el campo de los lenguajes de bases de datos relacionales. A mediados de los años setenta, el proyecto de IBM dio como resultado un primer lenguaje denominado SEQUEL (*Structured English Query Language*), que por razones legales se denominó más adelante SQL (*Structured Query Language*). Al final de la década de los setenta y al principio de la de los ochenta, una vez finalizado el proyecto System R, IBM y otras empresas empezaron a utilizar el SQL en sus SGBD relacionales, con lo que este lenguaje adquirió una gran popularidad.

En 1982, ANSI (*American National Standards Institute*) encargó a uno de sus comités (X3H2) la definición de un lenguaje de bases de datos relacionales. Este comité, después de evaluar diferentes lenguajes, y ante la aceptación comercial del SQL, eligió un lenguaje estándar que estaba basado en éste prácticamente en su totalidad. El SQL se convirtió oficialmente en el lenguaje estándar de ANSI en el año 1986, y de ISO (*International Standards Organization*) en 1987. También ha sido adoptado como lenguaje estándar por FIPS (*Federal Information Processing Standard*), Unix X/Open y SAA (*Systems Application Architecture*) de IBM.

En el año 1989, el estándar fue objeto de una revisión y una ampliación que dieron lugar al lenguaje que se conoce con el nombre de SQL1 o SQL89. En el año 1992 el estándar volvió a ser revisado y ampliado considerablemente para cubrir carencias de la versión anterior. Esta versión del SQL, que se conoce con el nombre de SQL2 o SQL92, es la que se utilizó en la escritura de la aplicación propuesta.

HTML

HyperText Markup Language (HTML), se utiliza para crear documentos que muestren una estructura de hipertexto. Permite, además, crear archivos de tipo multimedia, es decir, que contengan información más allá de lo textual, como imágenes, videos, sonidos y subprogramas activos como plug-ins y applets. (Mateu, 2004)

El lenguaje HTML es muy fácil de aprender. Ofrece una experiencia más agradable a los usuarios junto con otros lenguajes y tecnologías como JavaScript y CSS, que ayudan con la creación de estilos visuales, validación de datos y generación de contenidos dinámicamente. (Musciano y Kennedy, 1999)

1.7 Sistemas de gestión de bases de datos

Antes de conocer qué es un sistema de gestión de base de datos (SGBD), veamos qué se entiende por base de datos.

Una base de datos (BD) es un conjunto de datos que modelan hechos y objetos de una parcela de la realidad y sirven de soporte a una aplicación informática. Dichos datos deben estar almacenados físicamente en forma de ficheros informáticos y deben estar relacionados entre sí mediante una determinada estructura lógica. (Risco Nuñez, 2011)

Es posible considerar a la propia base de datos como una especie de armario electrónico para archivar; es decir, es un depósito o contenedor de una colección de archivos de datos computarizados. Los usuarios del sistema pueden realizar una variedad de operaciones sobre dichos archivos, por ejemplo:

- Agregar nuevos archivos vacíos a la base de datos.
- Insertar datos dentro de los archivos existentes.
- Recuperar datos de los archivos existentes.
- Modificar datos en archivos existentes.
- Eliminar datos de los archivos existentes.
- Eliminar archivos existentes de la base de datos.

El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos (SGBD). (Mato García, 2006)

Un sistema de gestión de base de datos es básicamente un sistema computarizado para guardar registros; es decir, es un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones. La información en cuestión puede ser cualquier cosa que sea de importancia para el individuo u organización; en otras palabras, todo lo que sea necesario para auxiliarle en el proceso general de su administración. (Date, 2001)

Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. (Mato García, 2006)

1.7.1 SQL Server

SQL Server es un sistema de gestión de bases de datos relacionales (SGBDR), compuesto por un conjunto de elementos, que se integran con el sistema operativo Windows NT, para proporcionar un entorno avanzado de procesamiento de datos, dentro de una arquitectura cliente-servidor. Fue desarrollado por Microsoft y permite, como su propio nombre indica, la gestión de un entorno de bases de datos relacional; se distribuye bajo una licencia propietaria. SQL Server abarca, tanto el área de diseño, como el de la de administración, proporcionando una interfaz bastante amigable con el usuario. (Sandoval, 2009)

SQL Server aprovecha las características multiproceso de Windows NT, utilizando todos los procesadores instalados para optimizar el manejo de datos. El sistema de seguridad de SQL Server está integrado con el de Windows NT. De esta forma, el usuario sólo debe identificarse al comenzar su sesión de trabajo con NT, puesto que al conectar con SQL Server, se establece una relación de confianza en la que SQL Server asume que si el usuario ha iniciado su sesión en el sistema, sus claves de acceso son correctas también para el motor de datos, por lo que realiza la conexión. (Sandoval, 2009)

1.7.2 PostgreSQL

PostgreSQL es un SGBD relacional orientada a objetos y libre, publicado bajo la licencia BSD (Berkeley Software Distribution). (Riveros, 2008)

PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre. (Riveros, 2008)

PostgreSQL posee una serie de características:

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos.

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arreglos (*arrays*).

Otras características

- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (*foreign keys*).
- Disparadores (*triggers*): Un disparador o *trigger* se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.

Las funciones son bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientada a objetos o la programación funcional. (The PostgreSQL Global Development Group, 2003)

PostgreSQL posee varios lenguajes para programar las funciones. Uno de estos lenguajes es el **PL/pgSQL**, muy parecido al PL/SQL de Oracle. (The PostgreSQL Global Development Group, 2003)

SQL es el lenguaje que PostgreSQL (y la mayoría del resto de bases de datos relacionales) usa como lenguaje de consultas. Es portable y fácil de aprender. Pero cada estamento SQL debe ser ejecutado individualmente por el servidor de bases de datos. Esto significa que su aplicación cliente debe enviar cada consulta al servidor de bases de datos, esperar a que se procese, recibir el resultado, realizar alguna computación, y luego enviar otras consultas al servidor. Todo esto incurre en una comunicación entre procesos y también puede sobrecargar la red si su cliente se encuentra en una máquina distinta al servidor de bases de datos. (The PostgreSQL Global Development Group, 2003)

Con PL/pgSQL puede agrupar un grupo de computaciones y una serie de consultas dentro del servidor de bases de datos, teniendo así la potencia de un lenguaje procedural y la sencillez de uso del SQL, pero ahorrando una gran cantidad de tiempo porque no tiene la sobrecarga de una comunicación cliente/servidor. Esto puede redundar en un considerable aumento del rendimiento. (The PostgreSQL Global Development Group, 2003)

PL/pgSQL añade a la potencia de un lenguaje procedural la flexibilidad y sencillez del SQL. Con PL/pgSQL puede usar todos los tipos de datos, columnas, operadores y funciones de SQL. (The PostgreSQL Global Development Group, 2003)

Debido a que las funciones PL/pgSQL corren dentro de PostgreSQL, estas funciones funcionarán en cualquier plataforma donde PostgreSQL corra. Así podrá reutilizar el código y reducir costes de desarrollo. (The PostgreSQL Global Development Group, 2003)

Entre las facilidades que brinda PostgreSQL podemos mencionar: la restauración continúa de la base de datos, es decir, puedes volver a un punto concreto. Es de suponer que esto representa una carga más para el sistema, pero es una opción interesante. Por otro lado ofrece mejoras de rendimiento y decisiones sobre el sistema de ficheros donde quieres guardar cosas, cambio de tipos de campo con ALTER TABLE, entre otras. PostgreSQL, no tiene costo asociado a la licencia del software. Ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento.

PostgreSQL tiene una legendaria confiabilidad y estabilidad y es fácil de administrar. (The PostgreSQL Global Development Group, 2003)

1.7.3 MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario, surgió alrededor de la década del 1990, creado por la empresa sueca MySQL AB. MySQL es un gestor

de base de datos sencillo de usar e increíblemente rápido. También es uno de los motores de base de datos más usados en Internet. (Riveros, 2008)

Es un software de código abierto, licenciado bajo la Licencia Pública General de GNU, aunque MySQL AB distribuye una versión comercial, en lo único que se diferencia de la versión libre, es en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de otra manera, se vulneraría la licencia GPL.

Características principales

Inicialmente, MySQL carecía de algunos elementos esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de esto, atrajo a los desarrolladores de páginas web con contenido dinámico, debido a su simplicidad, de tal manera que los elementos faltantes fueron complementados por la vía de las aplicaciones que la utilizan. Poco a poco estos elementos faltantes, están siendo incorporados tanto por desarrolladores internos, como por desarrolladores de software libre.

En las últimas versiones se pueden destacar las siguientes características principales:

- El principal objetivo de MySQL es velocidad y robustez.
- Soporta gran cantidad de tipos de datos para las columnas.
- Gran portabilidad entre sistemas, puede trabajar en distintas plataformas y sistemas operativos.
- Cada base de datos cuenta con 3 archivos, uno de estructura, uno de datos y uno de índice y soporta hasta 32 índices por tabla.
- Aprovecha la potencia de sistemas multiproceso, gracias a su implementación multihilo.
- Flexible sistema de contraseñas (passwords) y gestión de usuarios, con muy buen nivel de seguridad en los datos.
- El servidor soporta mensajes de error en distintos idiomas.
- Posee numerosos tipos de datos: enteros con y sin signo, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, y ENUM.
- Records de longitud fija y variable.
- Soporte de operadores y funciones completo en las cláusulas SELECT y WHERE de las consultas.
- Soporte para las cláusulas SQL: GROUP BY y ORDER BY. Además de funciones de agrupación (COUNT(), AVG(), STD(), SUM(), MAX(), MIN(), and GROUP_CONCAT()).
- Escrito en C y C++. Probado con un amplio rango de distintos compiladores.

Ventajas

- Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos.
- Facilidad de configuración e instalación.
- Soporta gran variedad de sistemas operativos.
- Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- Conectividad y seguridad.

Por las características mostradas anteriormente sumadas a la petición del cliente, se seleccionó MySQL como sistema de gestión de base de datos en la aplicación propuesta.

No obstante, en el proyecto se decidió utilizar un módulo de PHP, llamado **dbx**, que crea una capa de abstracción sobre la conexión entre la aplicación y el sistema de gestión de base de datos (SGBD) subyacente, lo que permite migrar la base de datos a otro SGBD sin tener que reescribir el código, solo modificando una línea en la configuración y convertir la BD existente puede permitir usar cualquiera de los SGBD mencionados anteriormente.

1.8 Herramientas de desarrollo

Las herramientas de desarrollo son aquellas que el programador usa para facilitar el proceso de escritura del software, su depuración y su prueba, así como trabajar en el diseño. En este proyecto se escogieron algunas de estas herramientas para maximizar la productividad y asistirse en el proceso del ciclo de vida de la aplicación: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

1.8.1 Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado (en inglés *Integrated Development Environment*) es un programa informático compuesto por un conjunto de herramientas de programación. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Puede

dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios. En esta aplicación se hará uso del IDE NetBeans en su versión 6.9.

NetBeans

El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, y está orientado principalmente a desarrollar en ese lenguaje, pero puede utilizarse para muchos otros lenguajes de programación. Existe además un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso y contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente. (NetBeans, 2011)

Este IDE realiza un análisis de los valores tomados por las variables en todo el proceso de ejecución. Es posible colocar puntos de parada (*breakpoints*) en los programas y realizar las acciones típicas de depuración. Por las características antes expuestas y por ser una herramienta que ofrece una amplia documentación y formación de recursos, se considera la herramienta ideal para llevar a cabo la programación del software cliente para Linux y con los módulos necesarios también es una excelente herramienta para el desarrollo de la aplicación web en PHP.

Geany

Geany es un editor de texto pequeño y ligero basado en Scintilla con características de entorno de desarrollo integrado (IDE).

Se desarrolló para proveer un IDE rápido y ligero, con muy poca dependencia de otros paquetes. Utiliza bibliotecas GTK+ para su funcionamiento. Está disponible para distintos sistemas operativos, como GNU/Linux, MacOS X, BSD, Solaris y Microsoft Windows.

Es extensible mediante un sistema de plug-ins, lo que le permite al usuario adicionar usabilidad y ajustarlo a su conveniencia.

Es distribuido como software libre bajo la Licencia Pública General de GNU. Debido a las anteriores funcionalidades, se utilizó como IDE en la programación del servidor en PHP y para el software cliente de las computadoras con Microsoft Windows.

1.8.2 Herramientas de modelado UML

Las herramientas CASE (Ingeniería de Software Asistida por Computadora), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (Centro de Estudios y Diseño de Sistemas, 2005)

El software de modelado UML ayuda a una construcción más rápida de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clase, código inverso, generar código desde diagramas y generar documentación.

Se escogieron las dos herramientas CASE descritas a continuación, puesto que el proyecto se diseñó e implementó en Linux, y se trabajó con software libre y open source , utilizando una para complementar la otra en caso que le faltara soporte para algún artefacto o para algún estándar UML.

ArgoUML

ArgoUML es una aplicación de modelado UML escrita en Java y publicada bajo la Licencia Pública de Eclipse, por lo que es software libre. Al estar escrita en Java, está disponible en cualquier plataforma soportada por el lenguaje (Microsoft Windows, Linux, MacOS X, etc.).

Sigue de cerca el estándar UML y soporta todos los diagramas de la versión 1.4. Además cuenta con soporte de internacionalización para alemán, chino, español, francés, inglés, italiano, portugués y ruso. Puede realizar ingeniería inversa sobre archivos JAR y es capaz de generar código en Java, PHP, Python, Ruby, C++ y Csharp (C#) desde los proyectos.

Puede exportar los diagramas en formatos PNG, GIF, JPG, SVG y EPS.

Por ser portable, de pocas dependencias y libre, es una alternativa viable como herramienta de modelado UML para Linux.

UMLet

UMLet es una herramienta UML escrita en Java. Liberada bajo la Licencia Pública General de GNU versión 3, es de código abierto y libre. Es ligera, portable y solo depende del entorno JRE.

Está diseñada para enseñar el Lenguaje de Modelado Unificado y para crear diagramas UML de manera rápida. Es una herramienta de dibujo más que una herramienta de modelado que el usuario puede extender con elementos nuevos o modificar los elementos que trae por defecto. Se pueden crear plantillas y objetos para ajustar la aplicación a las necesidades específicas de cada usuario, lo que requiere programarlos en Java.

Cuenta con una interfaz de usuario sencilla que usa código simple para modificar las formas básicas, su decoración y las anotaciones. Esto conlleva al usuario a aprender un lenguaje de etiquetado, pero con mínimo esfuerzo pues el lenguaje es fácil e intuitivo.

Puede exportar los diagramas a imágenes (JPG, PNG), dibujos (SVG, EPS) o documentos (PDF).

Los elementos UML más utilizados son soportados: clase, caso de uso, secuencia, estado, despliegue, actividad. No soporta UML 2.0 pero con la característica de creación y modificación de plantillas se puede obtener la misma funcionalidad.

Su formato nativo para salvar los proyectos es UXF, una extensión de XML dirigida a intercambiar modelos UML.

Se puede ejecutar en Linux, Windows, MacOS X o como un plug-in para el IDE Eclipse.

Esta herramienta es sencilla, pero para el usuario experimentado en UML es muy poderosa, permite crear diseños de manera rápida y eficiente, ajustados a un uso en particular.

1.9 Sistemas Operativos

Un sistema operativo (SO; también OS, del inglés *Operating System*) es el software básico de una computadora que provee una interfaz entre el resto de programas del ordenador, los dispositivos hardware y el usuario.

Se puede definir como un conjunto de programas que controlan directamente los recursos hardware o físicos de un ordenador (CPU, memoria principal y periféricos) proporcionando una máquina virtual más fácil de utilizar que el hardware subyacente. El sistema operativo es la capa de software más baja de un ordenador, controla todos los recursos de la computadora y establece la base sobre la que pueden escribirse los programas de aplicación.

En términos generales un sistema operativo se puede definir como el componente de software encargado de la interrelación de los programas con la máquina y de este conjunto con el usuario.

Los sistemas operativos realizan tareas básicas, tales como reconocimiento de la conexión del teclado, enviar la información a la pantalla, no perder de vista archivos y directorios en el disco, y controlar los dispositivos periféricos tales como impresoras, escáner, etc.

En sistemas grandes, el sistema operativo tiene incluso mayor responsabilidad y poder, es como un policía de tráfico, se asegura de que los programas y usuarios que están funcionando al mismo tiempo no interfieran entre ellos. El sistema operativo también es responsable de la seguridad, asegurándose de que los usuarios no autorizados no tengan acceso al sistema.

La mayoría de aparatos electrónicos que utilizan microprocesadores para funcionar, llevan incorporado un sistema operativo (teléfonos móviles, reproductores de DVD, computadoras, radios, enrutadores, televisores, etc.). En cuyo caso, son manejados mediante una interfaz gráfica de usuario, un gestor de ventanas o un entorno de escritorio si por ejemplo es un teléfono, una computadora o tableta; mediante una consola o control remoto si es un reproductor DVD o un televisor y mediante una línea de comandos o navegador web si es un enrutador.

1.9.1 Linux

Linux es un sistema operativo inspirado en Unix. Ensamblado bajo un modelo de desarrollo y distribución de código abierto, Linux es software libre. El componente definitorio de Linux es el núcleo del sistema operativo, liberado al mundo el 5 de octubre de 1991 por su autor Linus Torvalds.

Linux originalmente fue desarrollado como un sistema operativo libre para computadoras personales con arquitectura Intel x86, pero con el paso del tiempo ha sido llevado a muchas otras plataformas de hardware y se ha convertido en el más portable de los sistemas operativos.

Lidera los sectores de servidores, mainframes y supercomputadoras, donde sobre el 90% de las 500 supercomputadoras más rápidas del mundo lo ejecutan.

También es comúnmente instalado en sistemas compactos, en dispositivos donde el sistema operativo está instalado a muy bajo nivel, junto al firmware y altamente acoplado. Entre dichos sistemas encontramos teléfonos móviles, tabletas, enrutadores, cortafuegos, televisores, reproductores multimedia, consolas de video juegos, cajeros automáticos, cajas registradoras, AP inalámbricos, entre otros.

Android, el sistema operativo más utilizado en los teléfonos inteligentes, está construido sobre el núcleo de Linux.

El desarrollo de Linux es uno de los ejemplos más prominentes de colaboración de software de código abierto y libre. Su código fuente puede ser usado, modificado y distribuido comercialmente o sin costo alguno, bajo las cláusulas de la Licencia Pública General de GNU.

Una distribución Linux es una compilación de software basada en Linux que incluye determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios. Es la forma más común de encontrar un sistema operativo Linux y están orientados a diferentes requerimientos. Debido a la variedad del software libre surgen las más diversas ediciones, enfocadas al sector doméstico, empresarial y para servidores.

Existen distribuciones orientadas a escritorio, que por lo general incluyen un sistema gráfico, como Xorg y un entorno de escritorio, como Unity, GNOME o KDE Plasma. También incluyen aplicaciones como el navegador Mozilla Firefox, la compilación para office LibreOffice y GIMP como editor de imágenes.

Otras orientadas a servidores pueden por el contrario omitir todo entorno gráfico, hacer énfasis en aplicaciones de consola y administrativas, como los servidores Apache HTTP Server, OpenSSH, MySQL, PostgreSQL.

Algunas distribuciones populares son Debian, Ubuntu, Mint, Red Hat Entrerpirse Linux, Fedora, CentOS, Mandriva, OpenSUSE, Arch Linux, pero existen muchas otras; debido a la naturaleza de Linux todo usuario puede crear una distribución para cualquier uso que desee y por tanto, están en continuo surgimiento.

1.9.2 Microsoft Windows

Microsoft Windows (conocido generalmente como Windows), es el nombre de una familia de sistemas operativos desarrollados y vendidos por Microsoft, bajo la licencia Microsoft CLUF (una licencia EULA) por lo que es comercial y privativo. Microsoft introdujo un entorno operativo denominado Windows el 20 de noviembre de 1985 como una mejora sobre MS-DOS (*Microsoft Disk Operating System*) en respuesta al creciente interés en las interfaces gráficas de usuario (GUI, *Graphical User Interface*). Luego de la primera versión, su popularidad creció y comenzó a utilizarse de forma generalizada gracias a su GUI, sustituyendo al sistema operativo más extendido hasta el momento, MS-DOS, cuya interfaz consistía en una línea de comandos.

Microsoft Windows ha llegado a dominar el mercado mundial de computadoras personales, con más del 90% de la cuota de mercado.

Las versiones más recientes de Windows son Windows 8.1 y Windows 8 para equipos de escritorio, Windows Server 2012 para servidores y Windows Phone 8 para dispositivos móviles.

A pesar de un movimiento en favor de migrar las computadoras del país que ejecutan el sistema operativo Microsoft Windows a Linux, hay resistencia al cambio y la mayoría todavía tienen instalado el sistema operativo privativo y comercial de manera ilegal, con copias piratas.

Conclusiones parciales del capítulo 1

En este capítulo se realizó una revisión bibliografía para conocer los conceptos fundamentales de la investigación, se estudiaron las principales herramientas para elaborar el software y atendiendo a sus características y las particularidades del negocio se seleccionaron las siguientes:

- RUP (Proceso Unificado de Desarrollo) como la metodología más apropiada para el desarrollo del proyecto.
- UML como el lenguaje de modelación utilizado.
- ArgoUML 0.34 y UMLet 12.2 como herramientas de modelado UML.
- Se implementa un modelo cliente-servidor, donde el software cliente encuesta el hardware de cada computadora y la envía al servidor donde se almacena, además la aplicación web en el servidor brinda la posibilidad de gestión de toda la información al jefe de seguridad informática.
- Para la implementación del software cliente:
    - El lenguaje Java para sistemas operativos Linux, utilizando como herramienta de desarrollo el IDE NetBeans 6.9.
    - El lenguaje JavaScript en su dialecto JScript, para sistemas operativos Microsoft Windows, utilizando como herramienta de desarrollo el Geany 0.21.
- Para la implementación del servidor: los lenguajes HTML, PHP y SQL, utilizando como herramientas de desarrollo Geany y NetBeans.
- MySQL como sistema de gestión de bases de datos y phpMyAdmin como cliente para el manejo de las BD.

La selección de estas tecnologías se basó principalmente en el software libre y la característica de integrarse para funcionar en cualquier plataforma.

Capítulo 2. Descripción del software propuesto para la gestión de inventarios de hardware en la Universidad de Sancti Spíritus José Martí Pérez.

Introducción

En este capítulo se realiza un estudio del modelo del negocio, este proceso permite una mejor comprensión de la problemática al conocer cuáles son los actores del negocio, los trabajadores, los casos de uso, entre otros. Además se realizan diagramas de actividades, modelos de objeto y expansiones a los casos de uso, los cuales esclarecen el proceso a automatizar y los pasos que no se deben obviar.

Además en este capítulo se dejan plasmados los casos de uso del sistema, sus actores, diagramas de casos de uso del sistema así como la descripción de cada uno según lo propuesto por la metodología RUP.

2.1 Modelo del negocio

El modelado de negocio es una técnica para comprender los procesos de negocio de la organización. Además de identificar los casos de uso y las entidades del negocio relevantes que el software debe soportar, de forma que se puede modelar solo lo necesario para que se comprenda el contexto. (Jacobson, Booch, & Rumbaugh, 2000)

El modelo del negocio está soportado por dos tipos de modelos de UML: modelo de casos de uso y modelo de objetos. (Jacobson, Booch, & Rumbaugh, 2000) Seguidamente se describe el proceso de negocio que se lleva a cabo en la Uniss, mediante los artefactos propuestos por la metodología RUP y modelados por el lenguaje UML.

2.1.1 Identificación de los procesos del negocio

Cuando se hable de procesos de negocio se puede decir que son un grupo de tareas relacionadas lógicamente que se llevan a cabo en una determinada secuencia y forma, y que emplean los recursos de la organización para dar resultados que apoyen sus objetivos. (Rumbaugh, Booch, & Jacobson, 2006)

A partir del proceso planteado anteriormente se identificó el siguiente proceso de negocio: En la Universidad de Sancti Spíritus José Martí Pérez las computadoras cuentan con una ficha técnica que guarda las características del hardware que las conforman. Estas fichas técnicas son

responsabilidad del encargado de la seguridad informática de cada área, tanto mantenerlas almacenadas, como actualizarlas. A su vez, el jefe de la seguridad informática de la Uniss se encarga de mantener las fichas técnicas de toda la universidad, y de pedirlas a los encargados de área cuando sea necesario. Cuando los directivos de la Uniss necesitan la información del estado de las computadoras (su inventario de hardware), lo piden al jefe de seguridad informática, que a su vez recoge la información de los encargados por cada área de la universidad, y cuando cuenta con toda la información actualizada, conforma el reporte de todo el centro. Las computadoras cuentan con sellos de seguridad que solo pueden ser removidos por trabajadores especialistas autorizados y en presencia del jefe de seguridad informática o encargado del área.

2.1.2 Reglas del negocio

Las reglas del negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto de negocio. (Rumbaugh, Booch, & Jacobson, 2006)

Partiendo de lo planteado anteriormente se identificaron las siguientes reglas:

- Cada sistema de cómputo debe tener una ficha técnica, con la información actualizada de la información del hardware que la compone.
- Cada computadora tiene un responsable de ella, que actualiza la ficha técnica y el encargado de la seguridad informática del área se encarga de almacenar la información para cuando sea requerida.
- El jefe de seguridad informática mantiene el control de la información del estado de hardware de toda la universidad y realiza los reportes detallados para aquel directivo que lo necesite, o para otro uso, por ejemplo para conformar el plan de seguridad informática.
- Cada vez que se abra una computadora se debe tener constancia de la acción realizada en la misma, así como volver a colocar el sello.
- Los sellos de seguridad de las computadoras solo pueden ser removidos por los técnicos autorizados en presencia del jefe de seguridad informática.

2.1.3 Modelo de casos de uso del negocio

El modelo de Casos de Uso del Negocio es el encargado de describir los procesos de una empresa utilizando los casos de uso y los actores, en correspondencia, a su vez, con los procesos del negocio y los clientes. (Jacobson, Booch, & Rumbaugh, 2000)

Se define a través de: el diagrama de casos de uso, la descripción y el diagrama de actividades de los casos de uso.

2.1.4 Actores del negocio

Se considera actor del negocio a cualquier individuo, grupo, entidad, organización, máquina o sistema de información externo; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados. (Jacobson, Booch, & Rumbaugh, 2000)

Actor	Descripción
Directivo de la Uniss	Es el encargado de solicitar información acerca del inventario de hardware de las computadoras

Tabla 1: Actores del negocio

2.1.5 Diagrama de casos de uso del negocio

Los diagramas de casos de uso se utilizan para especificar las funcionalidades y el comportamiento de un sistema mediante su interacción con los usuarios y otros sistemas. O sea es un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. (Jacobson, Booch, & Rumbaugh, 2000)

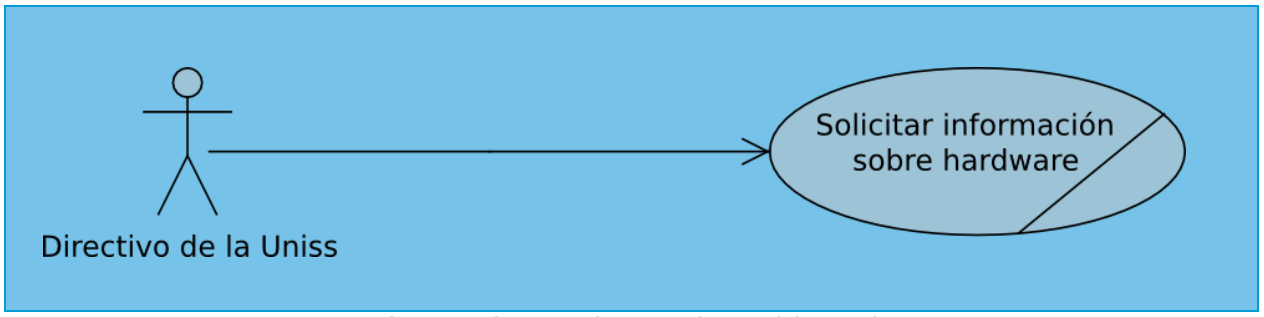

Figura 1: Diagrama de casos de uso del negocio

2.1.6 Trabajadores del negocio

Un trabajador es una abstracción de una persona o grupo de personas, una máquina o un sistema automatizado; que actúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio, manipulando entidades del mismo y representando un rol. (González, 2005)

Los trabajadores identificados son los siguientes:

Trabajador	Descripción
Jefe de seguridad informática de la Uniss	Es el responsable de solicitar la información de hardware a los encargados de la seguridad informática de las áreas para confeccionar el plan de seguridad informática.
Encargado de la seguridad informática del área	Es el responsable de confeccionar el reporte del inventario de hardware.

Tabla 2: Trabajadores del negocio

2.1.7 Descripción de los casos de uso del negocio

Actores del negocio:	**Directivo de la Uniss**
Propósito:	Obtener información sobre el hardware de todas las computadoras de la universidad.
Resumen: El caso de uso comienza cuando el directivo de la Uniss (actor) le solicita al jefe de seguridad informática la información referente al estado de hardware de las computadoras de la universidad. El jefe de seguridad informática de la Uniss, les solicita a su vez esta información a los encargados de la seguridad informática de las áreas, para conformar un reporte detallado de la situación del hardware del centro, el cual le entrega al directivo de la Uniss, culminando así el caso de uso.	
Flujo de trabajo	
Acción del actor	**Respuesta del negocio**
1. Los directivos de la Uniss le solicita al jefe de seguridad informática la información referente al estado de hardware.	2. El jefe de seguridad informática de la Uniss le solicita a los diferentes encargados de la seguridad informática por área que le conformen un informe de la situación del hardware de las computadoras de sus respectivas áreas. 3. El responsable de la seguridad informática del área realiza un chequeo de todas las computadoras bajo su control y

5. Los directivos reciben el reporte general del estado de hardware con la información solicitada.	conforma el informe del estado de hardware de área y lo entrega al jefe de seguridad informática. 4. El jefe de seguridad informática conforma el reporte general del estado de hardware de toda la Uniss y lo entrega al directivo que lo solicitó.
Mejoras: Cuando se pida la información el jefe de la seguridad informática no necesitará extender la solicitud a los responsables de área, él mismo directamente podrá acceder a la información actualizada de último momento desde cualquier estación de trabajo que esté conectada a la red de la Uniss.	

Tabla 3: Caso de uso: "Solicitar información sobre el hardware de las computadoras"

2.1.8 Diagramas de actividades

Un diagrama de actividades es un diagrama que muestra el flujo de actividad a actividad; los diagramas de actividad tratan la vista dinámica de un sistema. Un caso especial de diagrama de estados (aquellos diagramas que tratan la vista dinámica de un sistema) en el cual todos o casi todos los estados son estados de acción y en el cual todas o casi todas las transiciones son disparadas por la terminación de las acciones en los estados origen. (Rumbaugh, Booch, & Jacobson, 2006)

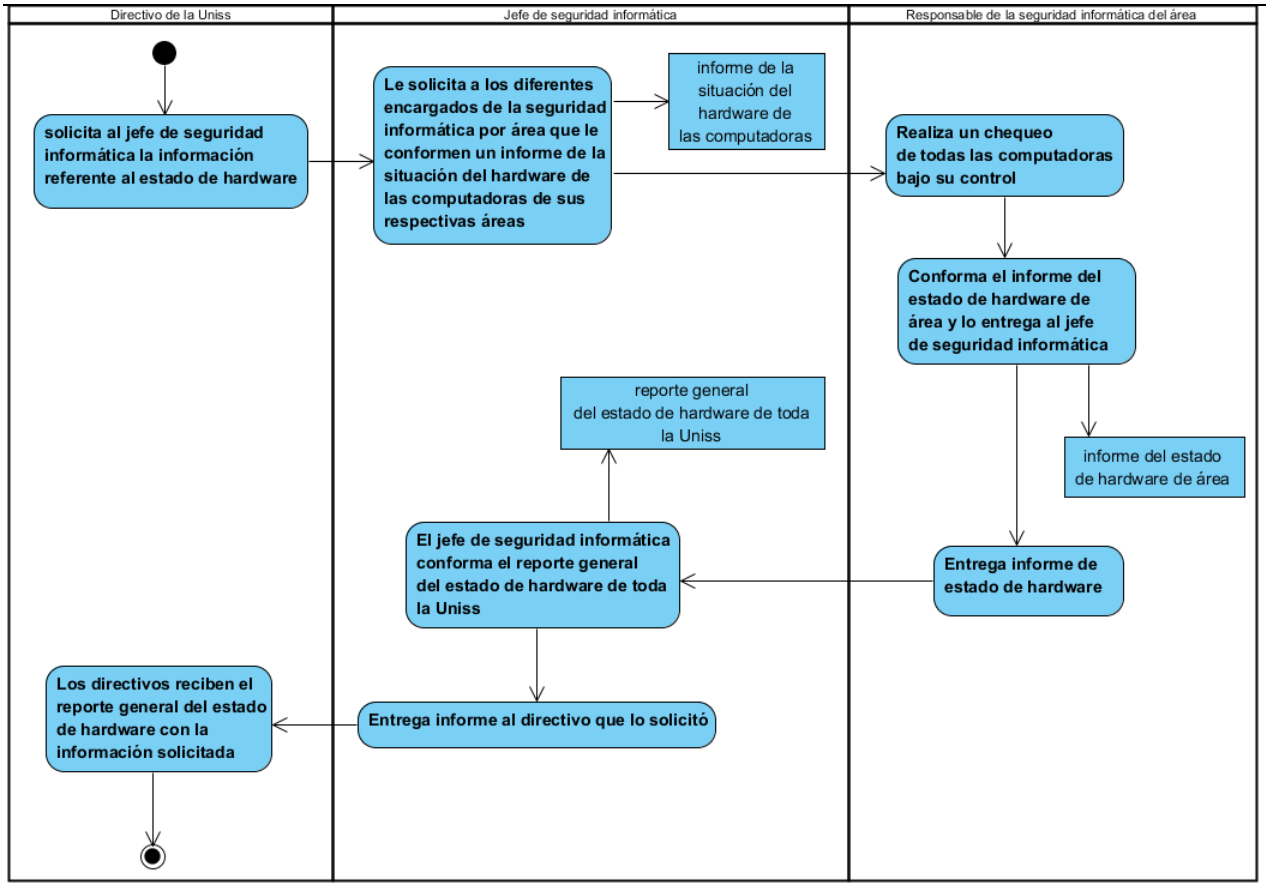

Figura 2: Diagrama de actividad

2.1.9 Modelo de objetos del negocio

El modelo de objetos del negocio se utiliza para describir la participación de los trabajadores y entidades del negocio, y su colaboración en la realización del negocio. Un modelo de objetos del negocio es un modelo interno a un negocio. Describe como cada caso de uso del negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y unidades de trabajo. (González, 2005)

Una entidad del negocio es algo que los trabajadores toman, inspeccionan, manipulan o producen en un caso de uso del negocio. (González, 2005)

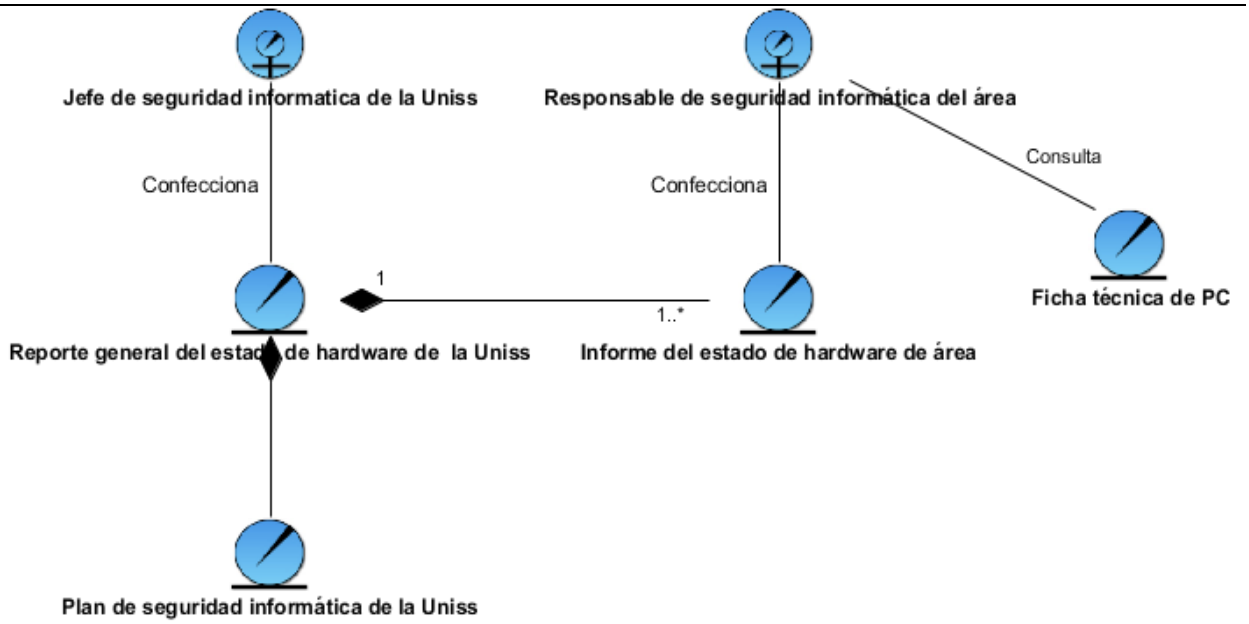

Figura 3: Modelo de objetos del negocio

2.2 Requerimientos

2.2.1 Requerimientos funcionales

Un requerimiento funcional expresa una especificación más detallada de las responsabilidades del sistema que se propone. Ellos permiten determinar, de una manera clara lo que debe hacer el sistema, siempre basándose en las necesidades de los usuarios. (González, 2005)

R1: Autentificar usuario

R2: Cambiar contraseña

R3: Mostrar computadoras registradas

3.1: Buscar computadora
3.2: Mostrar detalles de los componentes de hardware, periféricos y monitores
3.3: Mostrar detalles del software instalado

R4: Mostrar categorías de componentes

4.1: Mostrar detalles de los componentes de la computadora
4.2: Mostrar estado de los periféricos de la computadora

R5: Rastrear cambios de hardware

5.1: Escoger el tipo de dispositivo a rastrear

5.2: Escoger el tipo de cambio

5.3: Escoger fechas de inicio y de término para la ventana de tiempo

R6: Mostrar historial de cambios

6.1: Buscar computadora

6.2: Mostrar componentes de hardware

6.3 Mostrar historial de cambios de cada componente de hardware

R7: Búsqueda avanzada

7.1: Escoger componente a buscar

7.2: Escoger condición lógica

7.3: Escoger patrón de búsqueda

7.4: Adicionar filtro de búsqueda

R8: Mostrar estructura de la base de datos

R9: Mostrar software instalado

R10: Gestionar detalles de computadoras

8.1: Buscar computadora

8.2: Mostrar detalles de los componentes de hardware

8.4: Modificar los detalles de los componentes de hardware

R11: Cambiar estado de las computadoras registradas

R12: Cambiar estado de los componentes registrados

R13: Ejecutar consulta SQL

R14: Gestionar periféricos

14.1: Adicionar periféricos

14.2: Modificar estado de los periféricos

R15: Gestionar dispositivos

15.1: Adicionar dispositivos

15.2: Modificar estado de los dispositivos

R16: Ejecutar software cliente de extracción de hardware

2.2.2 Requerimientos no funcionales

Los requisitos no funcionales especifican cualidades y propiedades del sistema, como restricciones del entorno o de la implementación, rendimientos, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad. (Jacobson, Booch, & Rumbaugh, 2006)

Partiendo del criterio anterior, se especifican los siguientes requisitos no funcionales a tener en cuenta en el desarrollo de la aplicación:

Requerimientos de Apariencia o Interfaz Externa

- El software debe brindar una interfaz sencilla que facilite la interacción del usuario con el mismo, para maximizar la productividad y el tiempo de respuesta del usuario ante un requerimiento.
- La interfaz estará diseñada de modo tal que el usuario pueda tener en todo momento el control de la aplicación, lo que le permitirá ir de un punto a otro dentro de ella con gran facilidad. Se cuidará que la aplicación sea lo más interactiva posible.
- El ambiente deberá ser con colores claros y fuentes fáciles de leer, sin obstrucción del sistema en lo que realmente le importa al usuario.

Requerimientos de Usabilidad

- El sistema será utilizado por el jefe de seguridad informática de la Uniss, que está involucrado con el proceso de gestión de los inventarios de hardware del centro, así como la conformación del plan de seguridad informática, y el reporte de la situación del hardware de las computadoras.

Requerimientos de Rendimiento

- La herramienta propuesta debe ser rápida y el tiempo de respuesta debe ser el mínimo posible, adecuado a la rapidez con que el cliente requiere la respuesta a su acción.

Requerimientos de Soporte

- Se requiere un servidor ejecutando un servicio HTTP y soporte para PHP, con un sistema de gestión de bases de datos relacional, MySQL fue el escogido para desarrollar la aplicación. Se documentará la aplicación para garantizar su soporte.
- Las pruebas del sistema se realizarán en el departamento de Ingeniería Informática de la Facultad de Ingeniería de la Uniss y en los laboratorios de computadoras de la facultad. Dichas pruebas permitirán evaluar en la práctica la funcionalidad y las ventajas de este nuevo producto. El sistema deberá dar las posibilidades a futuras mejoras y nuevas opciones que se le quieran incorporar.

Requerimientos de Portabilidad

- El software se desarrollará en Linux y el servidor debe ser capaz de ejecutarse en cualquier sistema operativo, siempre que tenga instalado un servidor HTTP con un módulo PHP y un servidor de base de datos, escogiéndose MySQL. El software cliente que recopila la información de hardware también debe ejecutarse en Linux y Microsoft Windows. El jefe de seguridad informática debe poder administrar el sistema desde cualquier sistema operativo, solo contando con un navegador web para conectarse al servidor.

Requerimientos de Seguridad

- Debe garantizar la conectividad e integridad de los datos almacenados a través de la red. Esto está garantizado por el sistema operativo.
- Debe garantizar la confidencialidad para proteger la información de acceso no autorizado. Esto estará garantizado por el sistema de gestión de bases de datos.
- El sistema propuesto debe garantizar la protección de la información de acceso no autorizado, utilizando la autentificación para garantizar el cumplimiento de esto.
- Solo el usuario previamente autentificado podrá acceder a la información, gestionarla o modificarla. Solo existe un usuario, el jefe de seguridad informática de la Uniss que es administrador, todos los demás usuarios serán denegados el acceso al sistema.
- La aplicación tendrá validaciones de la información para contribuir a la seguridad del sistema.

- La información sensible de usuario y contraseña se guardará en la base de datos de manera encriptada, adicionando otra capa de seguridad.

Requerimientos de Confiabilidad

- Garantía de un tratamiento adecuado de las excepciones y validación de las entradas del usuario.
- El sistema en caso de fallos debe garantizar que las pérdidas de información sean mínimas.

Requerimientos de Disponibilidad

- Se le garantizará al usuario el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no retrasen al usuario para obtener los datos deseados en un momento dado.

Requerimientos de Ayuda y Documentación en Línea

- El sistema contará con una ayuda que explicará de manera clara y detallada al usuario todas las funcionalidades del sistema.
- La ayuda guiará usuario por todas las funcionalidades, dando especial importancia a las más complejas, con el objetivo de que pueda explotar a fondo toda la usabilidad del sistema, como se espera de un usuario avanzado.
- La ayuda quedará conformada por una página web con hipervínculos e imágenes que le facilitará al usuario poder ir de un lugar a otro sin perderse.

Requerimientos de Software

- El sistema operativo del servidor podrá variar, mientras que ejecute servicios de HTTP con soporte para PHP y sistema de gestión de base de datos relacional. Se desarrollará en Linux con Apache HTTP Server y MySQL como SGBD, por lo cual serán las alternativas recomendadas.
- Las máquinas que usará el jefe de seguridad informática como clientes para conectarse al servidor necesitarán un navegador web, por ejemplo Mozilla Firefox, Google Chrome o Internet Explorer.

- Las computadoras que se auditarán por el software cliente necesitarán tener una Java Virtual Machine (JVM) versión 6 o superior si tienen Linux; y un navegador Internet Explorer 6 o superior que permita ejecutar contenido ActiveX si tienen Microsoft Windows.

Requerimientos de Hardware

Para la puesta en práctica del proyecto se requieren máquinas con los siguientes requisitos:

- Se requiere de una máquina que funcione como servidor, la cual debe tener como mínimo:
  - Un Procesador Pentium III con 1 GHz de frecuencia o superior.
  - 512 Mb de Memoria RAM.
  - 40 GB de Disco Duro.
- Las computadoras utilizadas por el jefe de seguridad informática para conectarse al servidor requerirán como mínimo:
  - Un Procesador Pentium III.
  - 256 Mb de Memoria RAM.
  - Deben estar conectadas en red con el servidor a través de una tarjeta de red de 100 Mbps.

Restricciones en el diseño y la implementación

- Se utilizarán herramientas de desarrollo que garanticen la calidad de todo el ciclo de desarrollo del producto.

2.3 Modelo del sistema

2.3.1 Modelo de casos de uso del sistema

El modelado de Casos de Uso es la técnica más simple y que emplean los desarrolladores de software para modelar los requisitos del sistema desde la perspectiva del usuario. El modelo de casos de uso consiste en actores y casos de uso. Los actores representan usuarios y otros

sistemas que interaccionan con el sistema y los casos de uso representan el comportamiento del sistema, los escenarios que el sistema atraviesa en respuesta a un estímulo desde un actor. (Popkin Software and System, 2005)

2.3.2 Actores del sistema

Un actor no es más que un rol que juega un usuario de Caso de Uso cuando interaccionan con estos casos de uso. Los actores representan a terceros fuera del sistema que colaboran con el mismo. Una vez que se han identificado los actores del sistema, se ha identificado el entorno externo del sistema. (Rumbaugh, Booch, & Jacobson, 2006)

Actores	Descripción
Jefe de seguridad informática	Es el encargado de administrar y gestionar toda la información del sistema. Una vez que el software cliente adicione la información de las computadoras, se encarga de modificar el estado de propiedad o de uso de las computadoras, dispositivos de hardware y periféricos. Así como actualizar números de serie, añadir periféricos en reserva y cambiarlos de computadora o lugar. También utiliza el sistema para acceder a toda la información de hardware y estado de las computadoras, dispositivos y periféricos, creando reportes detallados con la información requerida, haciendo búsquedas avanzadas por patrones o ver los cambios realizados en una computadora mediante su historial, o sobre los dispositivos con un rastreador de cambios por fecha. Puede ejecutar el software cliente en la computadora que se requiera adicionar al sistema de forma manual.

Tabla 4: Actores del sistema

2.3.3 Casos de uso del sistema

Cada forma en que los actores usan el sistema se representa con un Caso de Uso. Los Casos de Uso son “fragmentos” de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. Un Caso de Uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia. (Jacobson, Booch, & Rumbaugh, 2000)

Para este sistema se definieron los siguientes casos de uso:

1. Autentificar usuario

2. Cambiar contraseña

3. Mostrar computadoras registradas

4. Mostrar categorías de componentes

5. Rastrear cambios de hardware

6. Mostrar historial de cambios

7. Búsqueda avanzada

8. Mostrar estructura de la base de datos

9. Mostrar software instalado

10. Gestionar detalles de computadoras

11 Cambiar estado de las computadoras registradas

12. Cambiar estado de los componentes registrados

13. Ejecutar consulta SQL

14. Gestionar periféricos

15. Gestionar dispositivos

16. Ejecutar software cliente de extracción de hardware

2.3.4 Diagramas de casos de uso del sistema

El modelo de casos de uso permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema. Describe lo que hace el sistema para cada tipo de usuario y proporciona la entrada fundamental para el análisis, el diseño y las pruebas. (Jacobson, Booch, & Rumbaugh, 2006)

Para facilitar el trabajo con los casos de uso y la organización de los elementos se agrupan los casos de uso en paquetes. Sobre los paquetes del sistema Pressman expresa: Subdividir los casos de uso en paquetes resulta de mucha ayuda en la modelación de cualquier sistema

informático. Los paquetes son un mecanismo de organización de elementos que subdividen el modelo en otros más pequeños que colaboran entre sí. (Pressman, 2007)

Nombre del paquete	Funcionalidades	Criterio de agrupamiento	Actores
Seguridad	▪ Autentificar usuario ▪ Cambiar contraseña	Funcionalidades (Requerimientos Funcionales)	Jefe de seguridad informática
Reportes	▪ Mostrar computadoras registradas ▪ Mostrar categorías de componentes ▪ Rastrear cambios de hardware ▪ Mostrar historial de cambios ▪ Búsqueda avanzada ▪ Mostrar estructura de la base de datos ▪ Mostrar software instalado	Funcionalidades (Requerimientos Funcionales)	Jefe de seguridad informática
Gestión	▪ Gestionar detalles de computadoras ▪ Cambiar estado de las computadoras registradas ▪ Cambiar estado de los componentes registrados ▪ Ejecutar consulta SQL ▪ Gestionar periféricos ▪ Gestionar dispositivos ▪ Ejecutar software cliente de extracción de hardware	Funcionalidades (Requerimientos Funcionales)	Jefe de seguridad informática

Tabla 5: Descripción de los paquetes

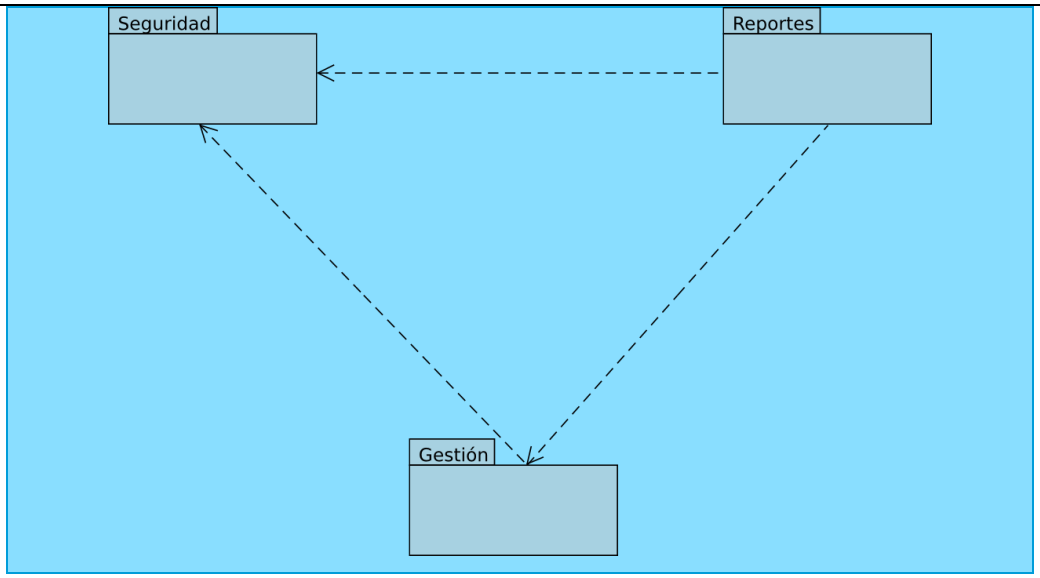

Figura 4: Diagrama de casos de uso por paquetes

Casos de uso: Paquete Seguridad

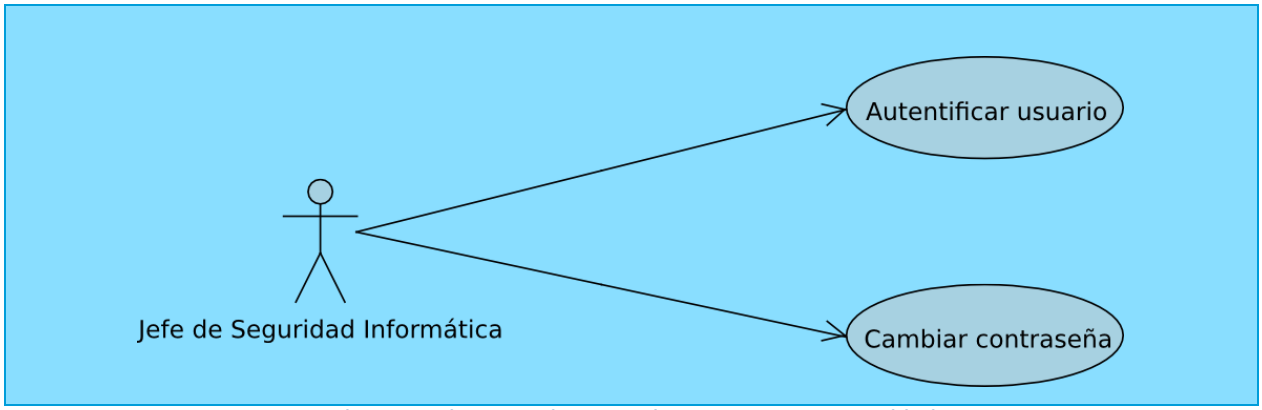

Figura 5: Diagrama de casos de uso: Paquete Seguridad

Descripción de los casos de uso: Paquete Seguridad

CU # 1	**Autentificar usuario.**
Actores:	Jefe de seguridad informática
Propósito:	Controlar el acceso al software.
Resumen:	

Comienza cuando el usuario solicita la entrada al software al introducir sus datos de usuario, si sus datos son correctos tiene acceso y control total sobre el software, en caso contrario se le informa a través de un mensaje de error y se le brinda la oportunidad de intentarlo de nuevo. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-1
Prototipo:	Anexo 3

Tabla 6: Descripción del caso de uso: Autentificar usuario

CU # 2	**Cambiar contraseña**
Actores:	Jefe de seguridad informática
Propósito:	Mantener la seguridad del sistema y la confiabilidad de los datos al ser la contraseña del usuario únicamente de su conocimiento.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de cambio de contraseña, una vez allí, introduce los datos necesarios: su contraseña actual y la nueva contraseña dos veces para validarla. Si todos los datos son correctos el sistema guarda la contraseña nueva, de lo contrario el sistema emite un mensaje de error explicando donde falló la acción, y se le brinda la oportunidad de intentarlo de nuevo. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-2
Prototipo:	Anexo 4

Tabla 7: Descripción del caso de uso: Cambiar contraseña

Casos de uso: Paquete Reportes

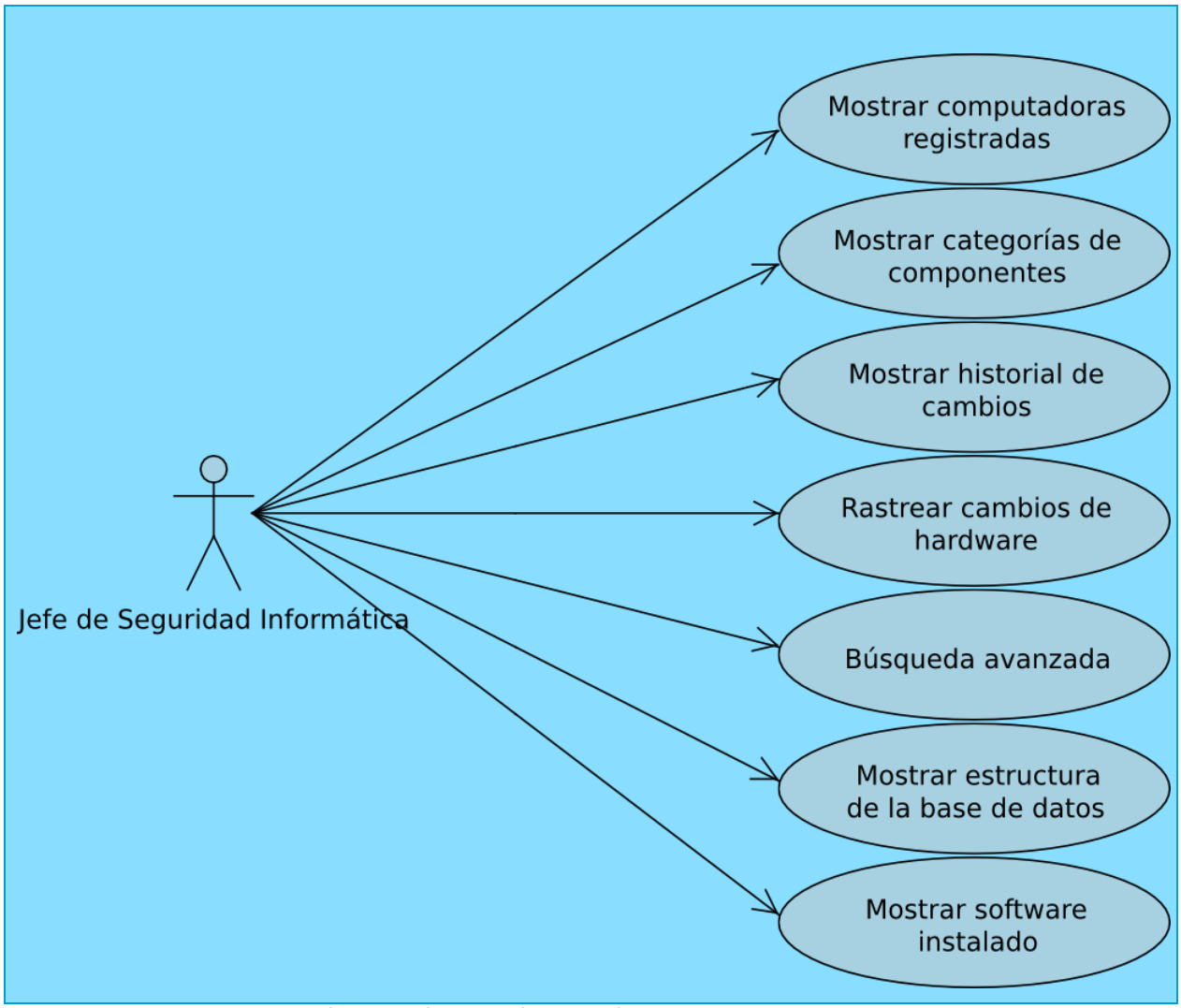

Figura 6: Diagrama de casos de uso: Paquete Reportes

Descripción de los casos de uso: Paquete Reportes

CU # 3	**Mostrar computadoras registradas**
Actores:	Jefe de seguridad informática

Propósito:	Mostrar información detallada de las computadoras registradas, así como de sus periféricos y monitores.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de mostrar computadoras registradas, el sistema le muestra una lista con todas las computadoras registradas y le brinda la opción de buscar por nombre y dominio, mostrándole las coincidencias encontradas o ninguna en caso de no existir. También puede escoger mostrar los periféricos y los monitores registrados. Con cada computadora encontrada se muestra un enlace a los detalles de hardware de la misma, donde están los datos completos de cada componente de hardware. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-3
Prototipo:	Anexo 5

Tabla 8: Descripción del caso de uso: Mostrar computadoras registradas

CU # 4	**Mostrar categorías de componentes**
Actores:	Jefe de seguridad informática
Propósito:	Mostrar las categorías de los componentes de hardware.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de mostrar categorías de componentes de hardware, el sistema le muestra una lista con todos los posibles componentes y periféricos, con un enlace en el nombre de cada tipo. El usuario escoge el tipo de componente que quiere ver y el sistema le muestra todos los componentes de ese tipo registrados, su cantidad y la PC donde están. También brinda la opción de buscar por nombre, mostrándole las coincidencias encontradas o ninguna en caso de no existir. Con cada coincidencia encontrada se muestra un enlace a los detalles de hardware de la misma, donde están los datos completos de cada componente de hardware de la PC donde está vinculado. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	

Referencias:	RF-4
Prototipo:	Anexo 6

Tabla 9: Descripción del caso de uso: Mostrar categoría de componentes

CU # 5	**Rastrear cambios de hardware**
Actores:	Jefe de seguridad informática
Propósito:	Rastrear los cambios realizados en el hardware.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de rastrear cambios de hardware, una vez allí el sistema le provee con todos los tipos de dispositivos de hardware posibles para escoger, también con los tipos de cambios que desea rastrear, que pueden ser adiciones, extracciones, o ambos, y por último con un intervalo de tiempo dentro del cual se harán las búsquedas de estos cambios; mostrándole las coincidencias encontradas o ninguna en caso de no existir. Con cada coincidencia encontrada se muestra un enlace a los detalles de la PC donde ocurrieron los cambios, así como los detalles de fechas exactas y detalles de hardware. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-5
Prototipo:	Anexo 7

Tabla 10: Descripción del caso de uso: Rastrear cambios de hardware

CU # 6	**Mostrar historial de cambios**
Actores:	Jefe de seguridad informática
Propósito:	Mostrar el historial de cambios de hardware de cada PC.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de mostrar computadoras registradas, el sistema le muestra una lista con todas las computadoras registradas y le brinda la opción de buscar por nombre y dominio, mostrándole las	

coincidencias encontradas o ninguna en caso de no existir. Con cada computadora encontrada se muestra un enlace al historial de cambios del hardware de dicha PC, donde se muestran todos los cambios que han ocurrido en cada componente de hardware por tipos, mostrando el tipo de cambio, adición, extracción, o modificación. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-6
Prototipo:	Anexo 8

Tabla 11: Descripción del caso de uso: Mostrar historial de cambios

CU # 7	**Búsqueda avanzada**
Actores:	Jefe de seguridad informática
Propósito:	Realizar una búsqueda avanzada.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de realizar búsqueda avanzada, una vez allí el sistema le provee con todos los tipos de dispositivos de hardware posibles para escoger, así como el tipo de filtro de búsqueda, también puede adicionar otros filtros haciendo la búsqueda más exhaustiva y específica, mostrándole las coincidencias encontradas o ninguna en caso de no existir. Con cada coincidencia encontrada se muestran los detalles del dispositivo, como fecha de ingreso, de extracción y cantidad, entre otros. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-7
Prototipo:	Anexo 9

Tabla 12: Descripción del caso de uso: Búsqueda avanzada

CU # 8	**Mostrar estructura de la base de datos**
Actores:	Jefe de seguridad informática
Propósito:	Mostrar la estructura de la base de datos.

Resumen: Comienza cuando el jefe de seguridad informática accede a la página de mostrar estructura de la base de datos, una vez allí el sistema muestra la estructura completa de forma detallada, con tipos de datos de los atributos, restricciones y relaciones entre tablas, culminando así el caso de uso.	
Referencias:	RF-8
Prototipo:	Anexo 10

Tabla 13: Descripción del caso de uso: Mostrar estructura de la base de datos

CU # 9	**Mostrar software instalado**
Actores:	Jefe de seguridad informática
Propósito:	Mostrar el software instalado en las PC.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de mostrar software instalado, el sistema muestra una lista con todas las computadoras registradas y le brinda la opción ver el software instalado en cada una. También brinda la opción de buscar por nombre de software, mostrándole las coincidencias encontradas o ninguna en caso de no existir. Con cada coincidencia encontrada se muestra un enlace a la computadora donde está instalado dicho software, donde se listan todos los instalados en dicha PC. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-9
Prototipo:	Anexo 11

Tabla 14: Descripción del caso de uso: Mostrar software instalado

Casos de uso: Paquete Gestión

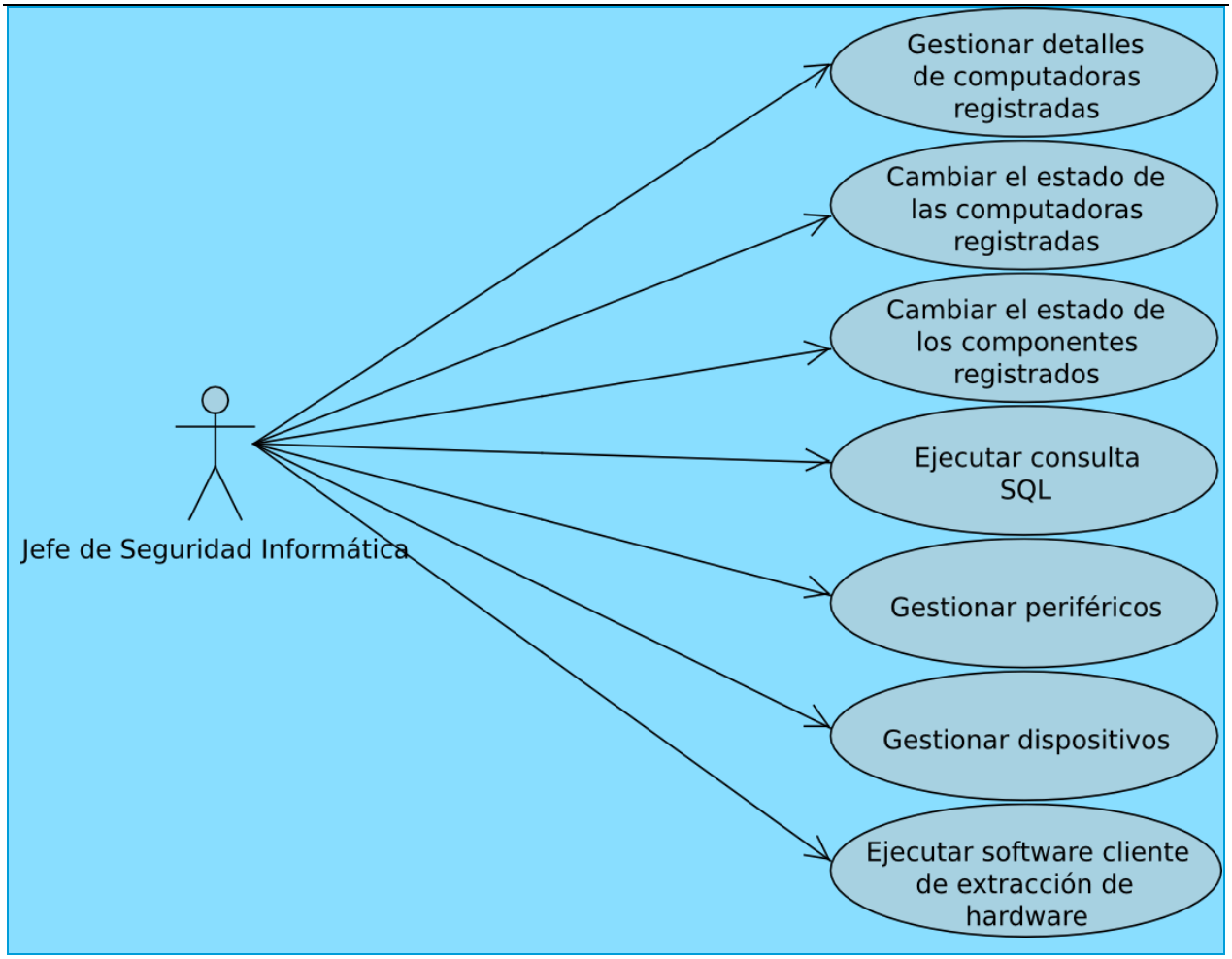

Figura 7: Diagrama de casos de uso: Paquete Gestión

Descripción de los casos de uso: Paquete Gestión

CU # 10	**Gestionar detalles de computadoras**
Actores:	Jefe de seguridad informática
Propósito:	Gestionar detalles de las computadoras registradas.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de mostrar computadoras registradas, el sistema le muestra una lista con todas las computadoras registradas y le brinda la opción de buscar por nombre y dominio, mostrándole las coincidencias encontradas o ninguna en caso de no existir. Con cada computadora	

encontrada se muestra un enlace a los detalles de hardware de la misma, donde está el enlace a gestionar los detalles de dicha PC, una vez allí el usuario puede adicionar periféricos o monitores, cambiar los detalles de los mismos, cambiar el estado de propiedad, el lugar donde se encuentra la PC o el número de inventario. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-10
Prototipo:	Anexo 12

Tabla 15: Descripción del caso de uso: Gestionar detalles de computadoras

CU # 11	**Cambiar estado de las computadoras registradas**
Actores:	Jefe de seguridad informática
Propósito:	**Cambiar estado de las computadoras registradas**.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de mostrar computadoras registradas, el sistema le muestra una lista con todas las computadoras registradas y le brinda la opción de buscar por nombre y dominio, mostrándole las coincidencias encontradas o ninguna en caso de no existir. Con cada computadora encontrada se muestra un enlace a los detalles de estado de la misma, una vez allí el usuario puede cambiar el estado de uso, puede estar en uso activo o en reserva (almacenada), también el estado de la propiedad, puede ser propia o puede ser prestada, si es prestada se puede escribir el lugar de origen, también se puede actualizar el lugar donde está emplazada actualmente. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-11
Prototipo:	Anexo 13

Tabla 16: Descripción del caso de uso: Cambiar estado de las computadoras registradas

CU # 12	Cambiar estado de los componentes registrados
Actores:	Jefe de seguridad informática
Propósito:	Cambiar estado de los componentes registrados.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de mostrar categorías de componentes de hardware, el sistema le muestra una lista con todos los posibles componentes y periféricos, con un enlace en el nombre de cada tipo. El usuario escoge el tipo de componente que quiere ver y el sistema le muestra todos los componentes de ese tipo registrados, su cantidad y la PC donde están. También brinda la opción de buscar por nombre, mostrándole las coincidencias encontradas o ninguna en caso de no existir. Con cada coincidencia encontrada se muestra un enlace a los detalles de estado de la misma, una vez allí el usuario puede cambiar el estado de uso, puede estar en uso activo o en reserva (almacenada), también el estado de la propiedad, puede ser propia o puede ser prestada, si es prestada se puede escribir el lugar de origen, también se puede actualizar el lugar donde está emplazada actualmente, también se puede modificar la computadora a la que está adjunto. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-12
Prototipo:	Anexo 14

Tabla 17: Descripción del caso de uso: Cambiar estado de los componentes registrados

CU # 13	Ejecutar consulta SQL
Actores:	Jefe de seguridad informática
Propósito:	Ejecutar una consulta SQL manualmente
Resumen: comienza cuando el jefe de seguridad informática desea ejecutar una consulta SQL manualmente para generar reportes personalizados y accede al menú de ejecución de consulta SQL, introduce la consulta y el sistema chequea que sea una consulta válida y no una prohibida, la ejecuta y muestra los datos resultantes, en caso contrario muestra	

un mensaje de error. El caso de uso finaliza cuando se ejecuta una de las acciones descritas previamente.	
Referencias:	RF-13
Prototipo:	Anexo 15

Tabla 18: Descripción del caso de uso: Ejecutar consulta SQL

CU # 14	**Gestionar periféricos**
Actores:	Jefe de seguridad informática
Propósito:	Gestionar los periféricos.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de gestionar periféricos, allí el sistema le muestra una forma que le permite adicionar periféricos o monitores nuevos, con su estado de uso y propiedad, así como el lugar donde están, si están prestados el lugar de origen, y la computadora a donde están conectados. También los números de serie, la opción de cambiar nombres y toda la información relevante de los mismos. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-14
Prototipo:	Anexo 16

Tabla 19: Descripción del caso de uso: Gestionar periféricos

CU # 15	**Gestionar dispositivos**
Actores:	Jefe de seguridad informática
Propósito:	Gestionar los dispositivos.
Resumen: Comienza cuando el jefe de seguridad informática accede a la página de gestionar dispositivos, el sistema brinda la opción de buscar por nombre, mostrándole las coincidencias encontradas o ninguna en caso de no existir. El sistema le muestra una forma	

que le permite modificar su estado de uso y propiedad, así como el lugar donde están, si están prestados el lugar de origen, y la computadora a donde están conectados. También los números de serie, la opción de cambiar nombres y toda la información relevante de los mismos. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-15
Prototipo:	Anexo 17

Tabla 20: Descripción del caso de uso: Gestionar dispositivos

CU # 16	**Ejecutar software cliente de extracción de hardware**
Actores:	Jefe de seguridad informática
Propósito:	Ejecutar software cliente de extracción de información de hardware de la PC.
Resumen: Comienza cuando el jefe de seguridad informática desea registrar la información de hardware de una computadora en el sistema y accede a la página de ejecución de software cliente. Una vez allí escoge el software a ejecutar basado en el sistema operativo de la PC a la cual se le va a realizar el inventario, si es Linux se descarga y ejecuta el software usando la JVM, y si es Microsoft Windows se ejecuta directamente desde el navegador. El caso de uso finaliza cuando se ejecuta una de las operaciones antes descrita.	
Referencias:	RF-16
Prototipo:	Anexo 18

Tabla 21: Descripción del caso de uso: Ejecutar software cliente de extracción de hardware

Conclusiones parciales del capítulo 2

Este capítulo deja claro cómo funciona el negocio a través de los artefactos proporcionados por la metodología RUP y las reglas del negocio, entre otros. Además contiene una descripción

general del sistema identificando los requerimientos funcionales y no funcionales y los casos de uso. La construcción de todos estos artefactos propició que:

- Se esclareciera cómo es el flujo de eventos que se realiza en cada uno de los casos de uso.
- Se establecieran las relaciones de cada uno de los actores del sistema con los casos de uso.

Capítulo 3. Construcción de la aplicación propuesta para la gestión de inventarios de hardware en la Universidad de Sancti Spíritus José Martí Pérez.

Introducción

La parte del proceso de desarrollo de software cuyo principal propósito es decidir cómo se llevará a cabo el sistema, es el diseño. Durante esta etapa se toman decisiones importantes que conllevan al efectivo cumplimiento de los requerimientos funcionales y la obtención de un software con calidad.

En el presente capítulo se plasman los resultados de la etapa del diseño e implementación del sistema, utilizando UML para su modelado. Se presentan los diagramas de clases del diseño, diagramas de clases persistentes y modelo de datos, diagrama de componente y de despliegue. Además se describen los principios de diseño aplicados determinando los estándares usados en la interfaz de la aplicación, la concepción general de la ayuda, el tratamiento de errores, cómo es manejada la seguridad y el tratamiento de los estándares de codificación.

3.1 Diagrama de clases del diseño

Una clase de diseño es una abstracción de una clase o construcción similar en la implementación del sistema. (Jacobson, Booch, & Rumbaugh, 2000)

Un diagrama de clases es una colección de elementos declaratorios del modelo, como clases, tipos y sus relaciones; conectados unos a otros y a sus contenidos en forma de grafo. (Jacobson, Booch, & Rumbaugh, 2000)

Los diagramas de clases del diseño fueron elaborados a partir de los diferentes casos de uso del sistema. Los mismos se muestran a continuación.

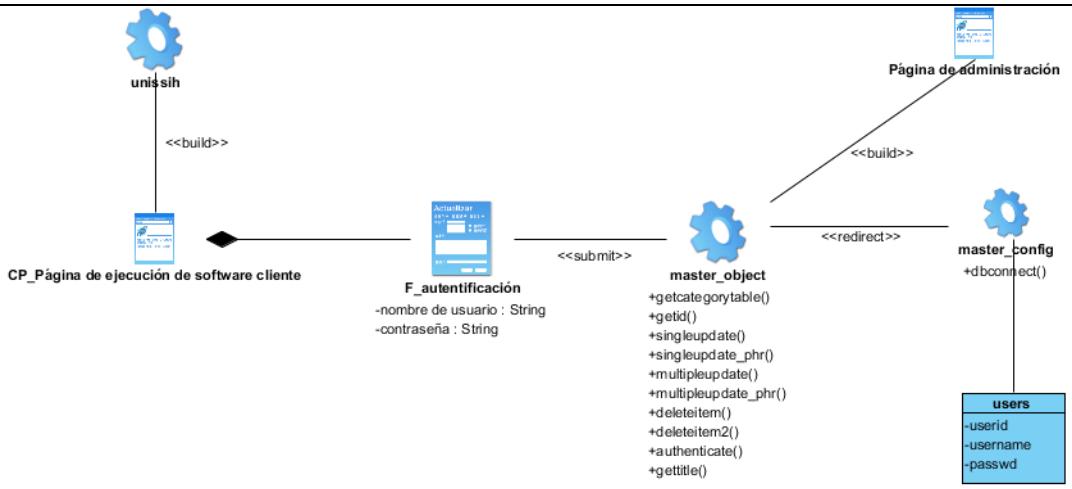

Figura 8: Diagrama de clases: Autentificar usuario

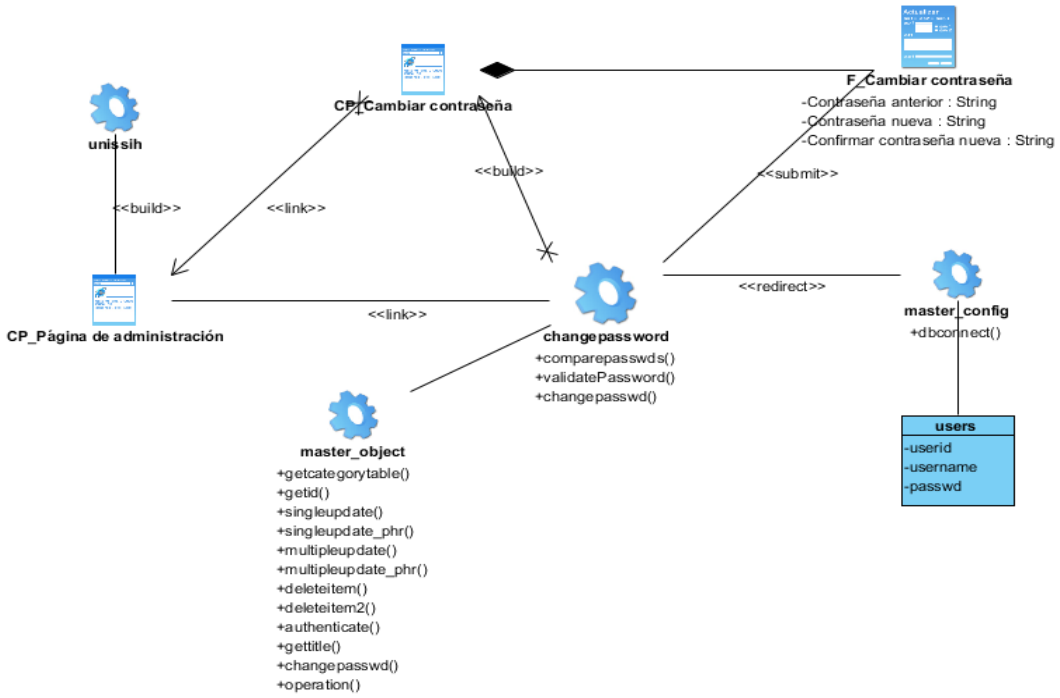

Figura 9: Diagrama de clases: Cambiar contraseña

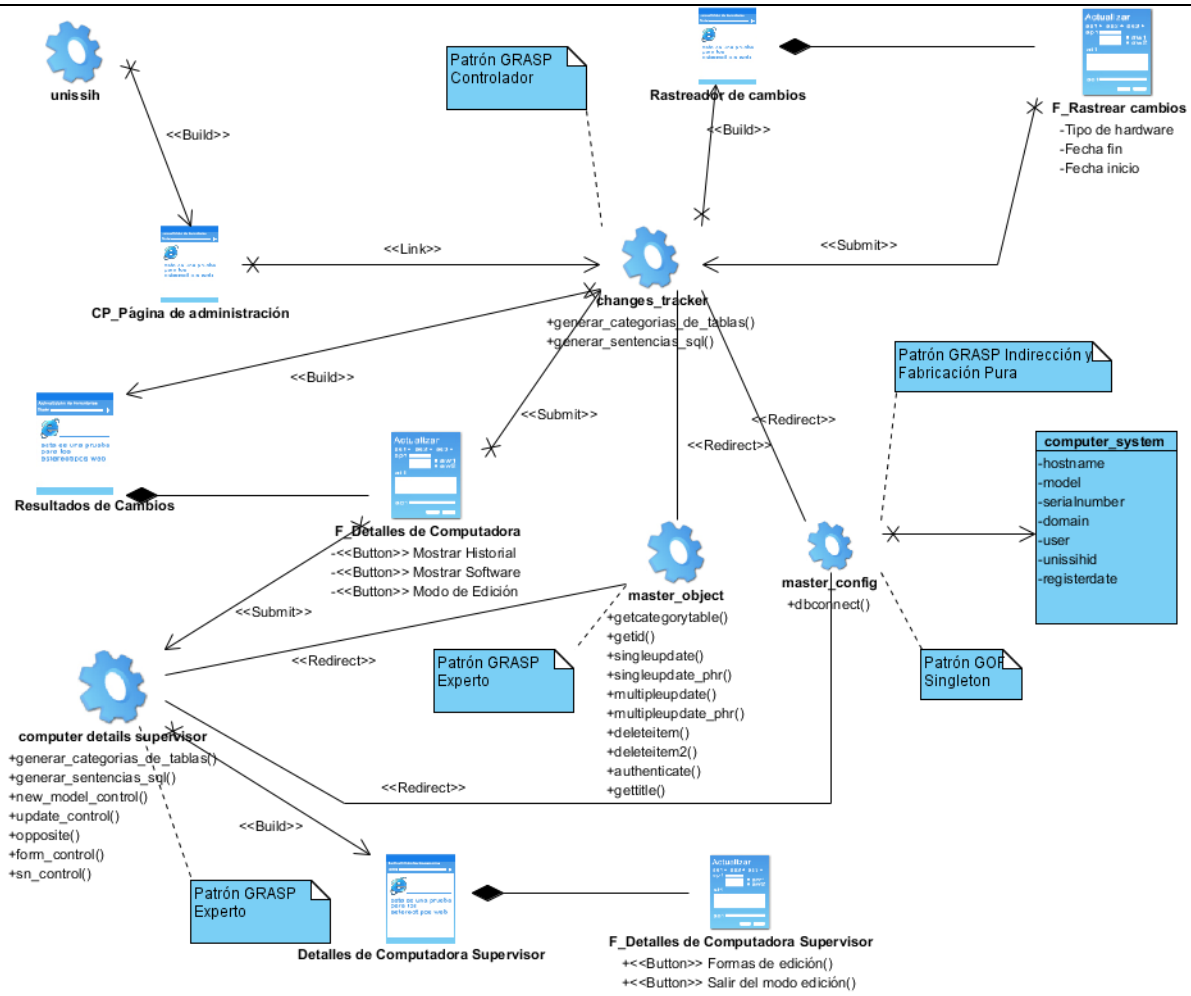

Figura 10: Diagrama de clases: Mostrar computadoras registradas

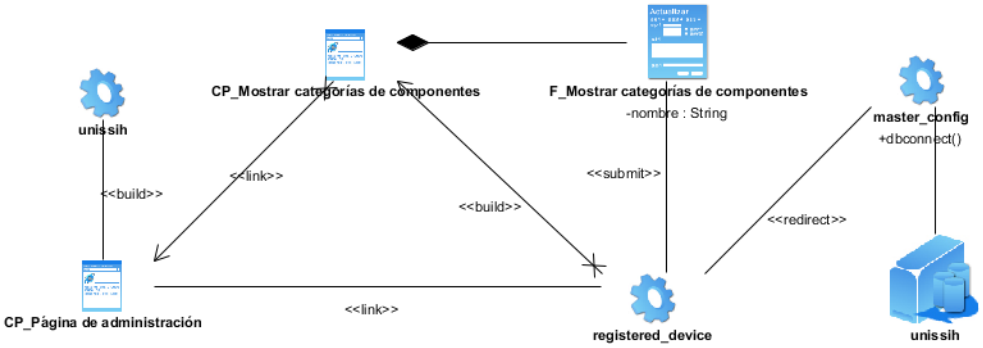

Figura 11: Diagrama de clases: Mostrar categoría de componentes

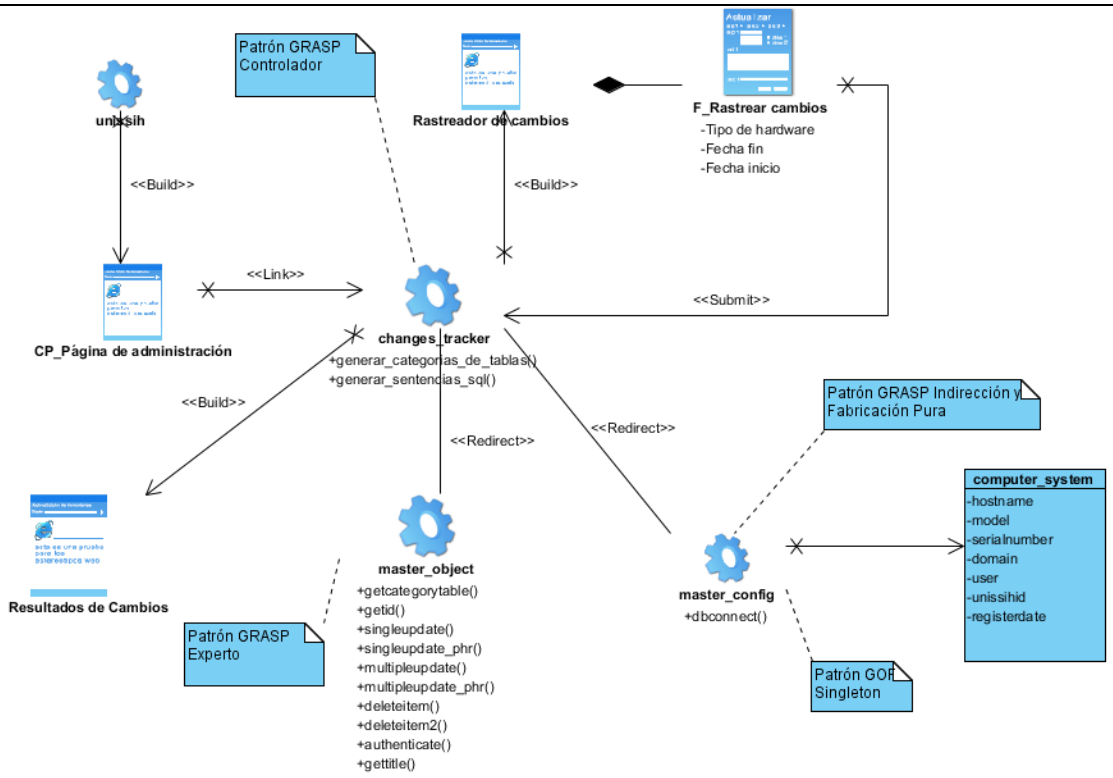

Figura 12: Diagrama de clases: Rastrear cambios de hardware

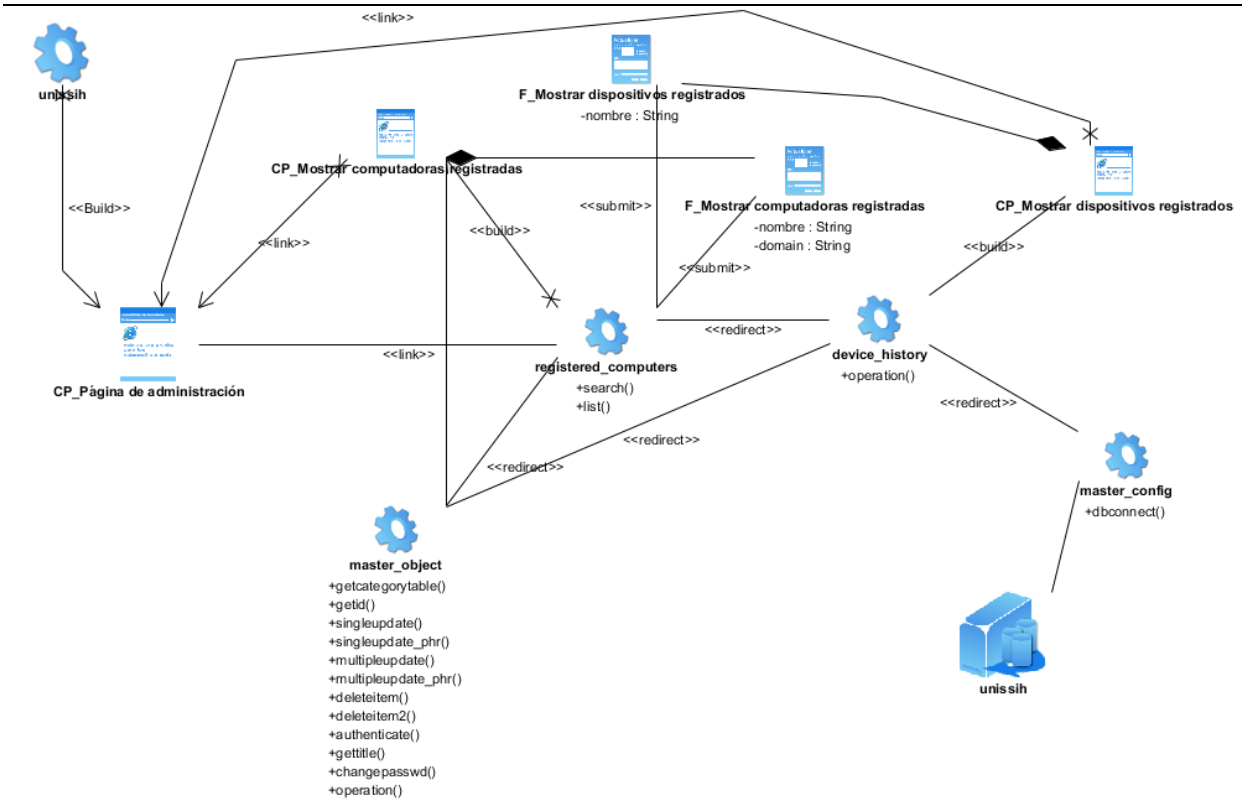

Figura 13: Diagrama de clases: Mostrar historial de cambios

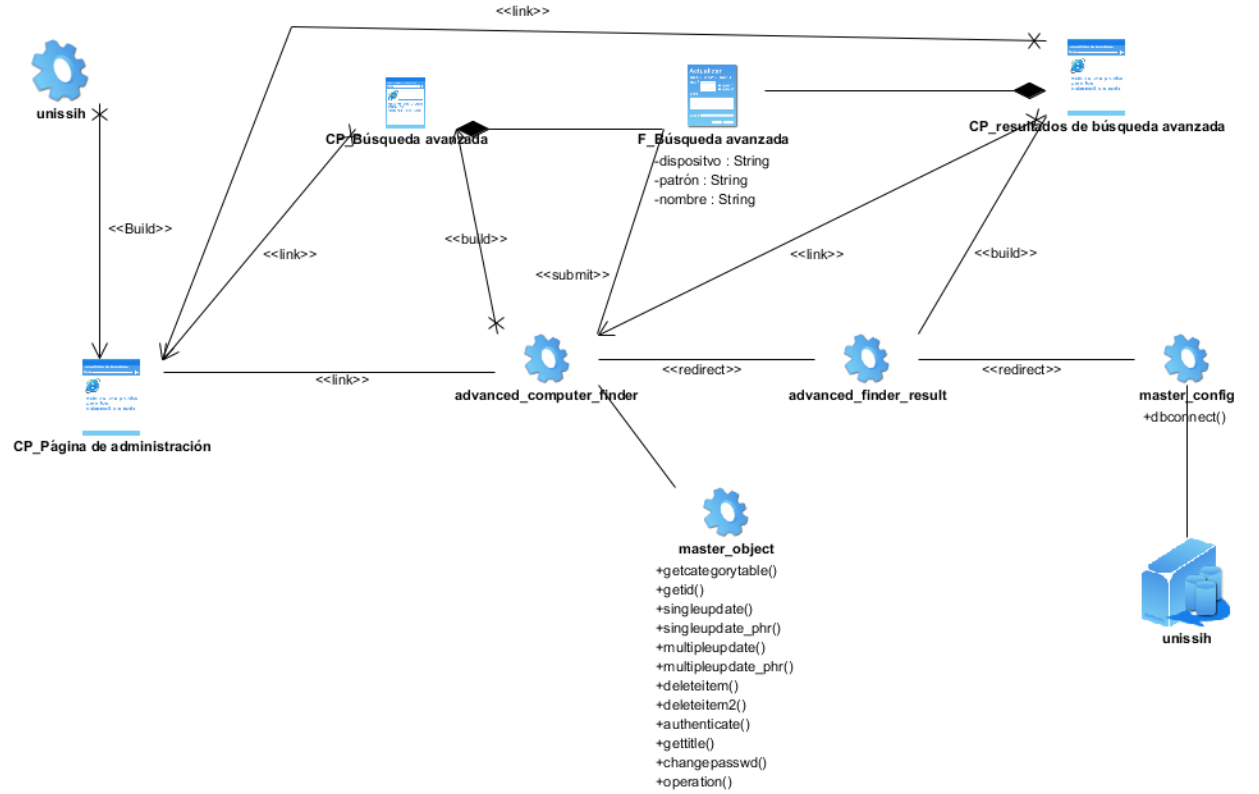

Figura 14: Diagrama de clases: Búsqueda avanzada

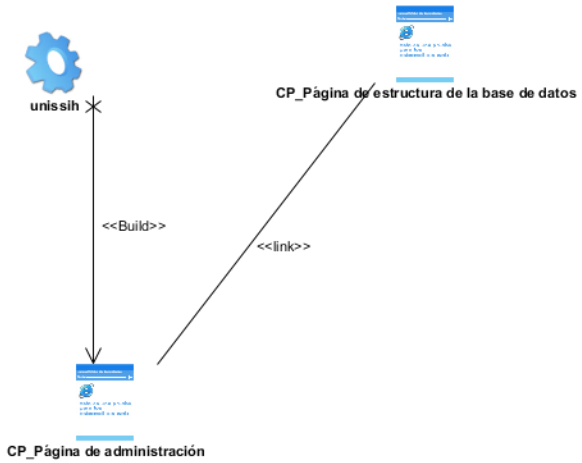

Figura 15: Diagrama de clases: Mostrar estructura de la base de datos

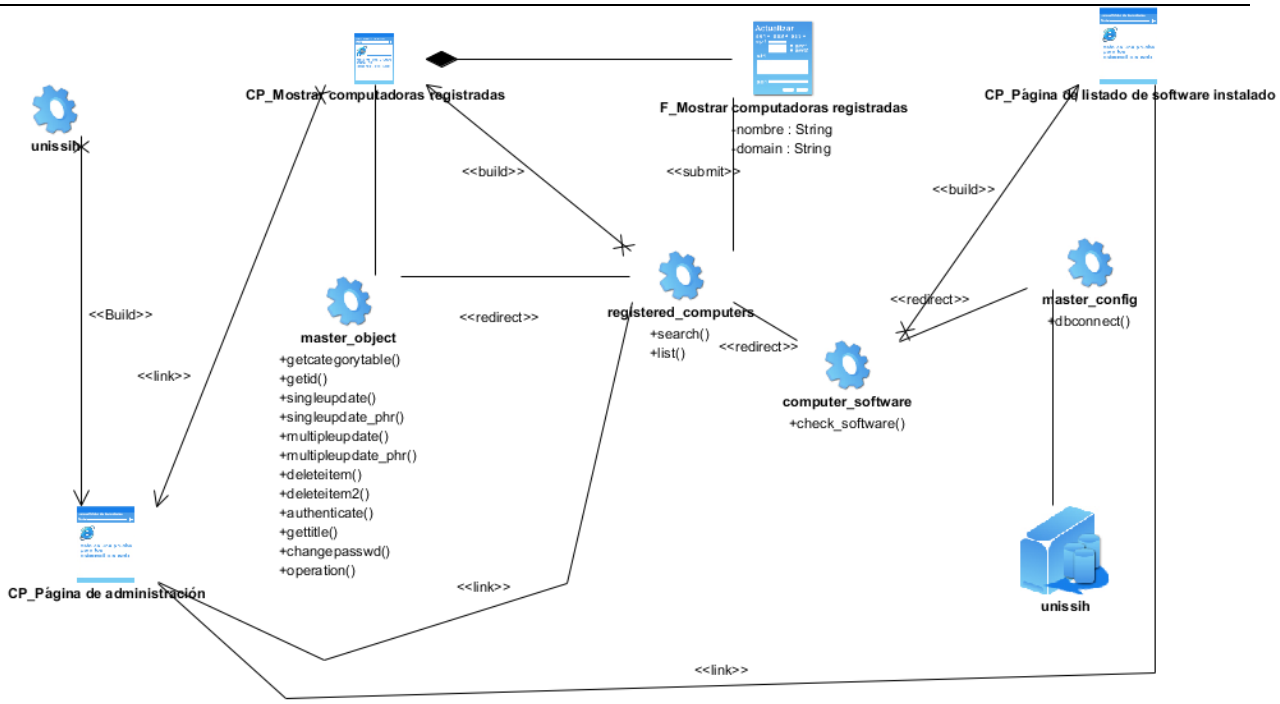

Figura 16: Diagrama de clases: Mostrar software instalado

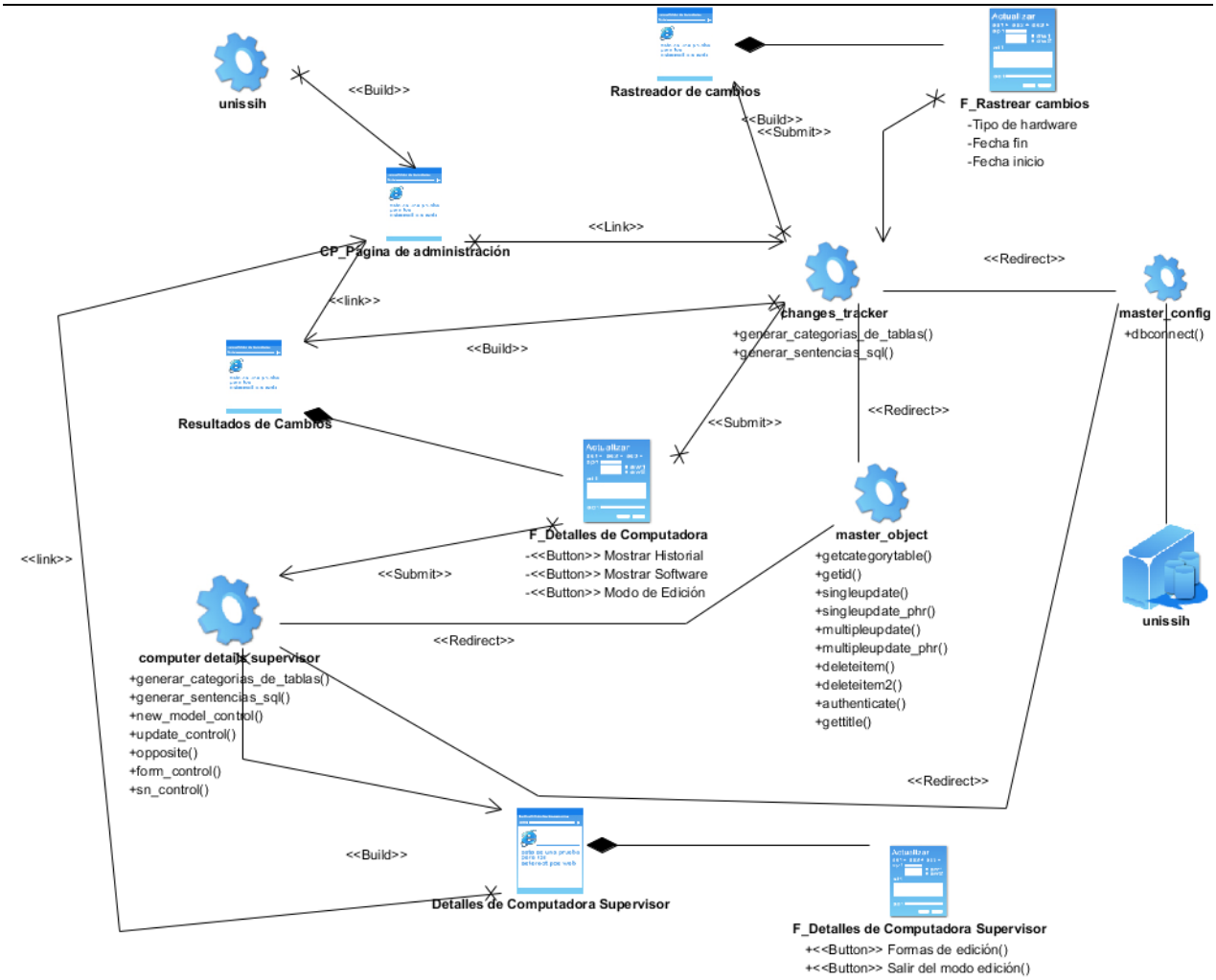

Figura 17: Gestionar detalles de computadoras

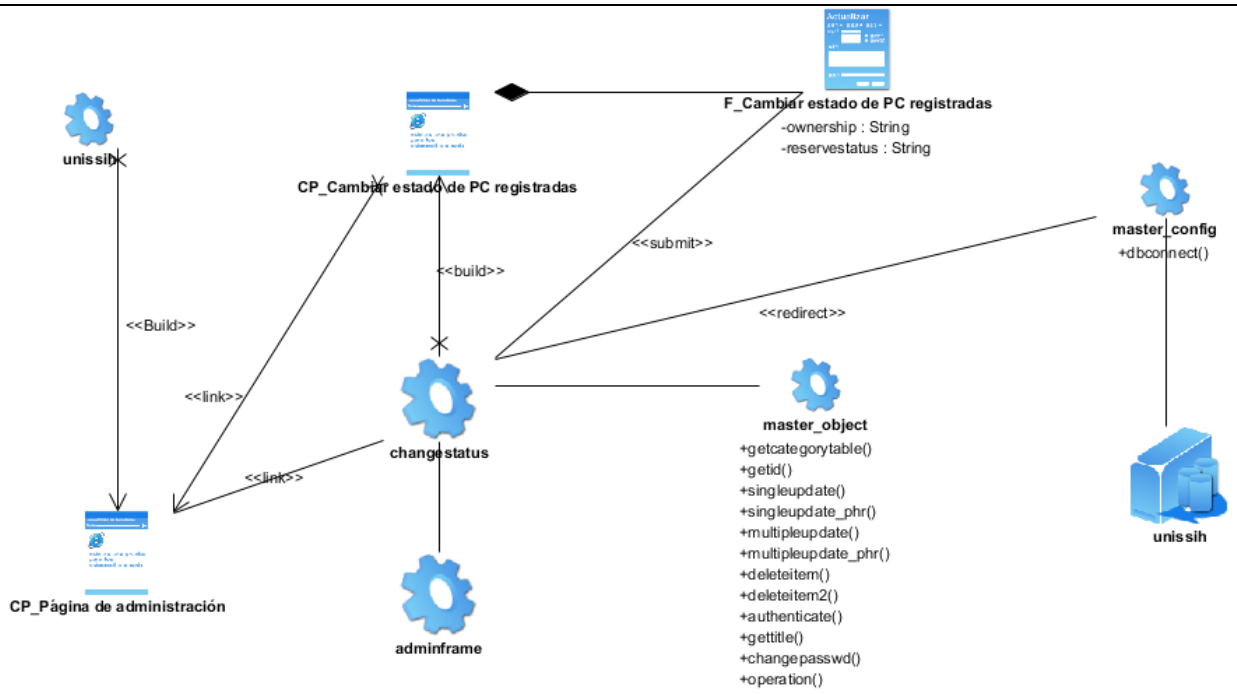

Figura 18: Cambiar estado de las computadoras registradas

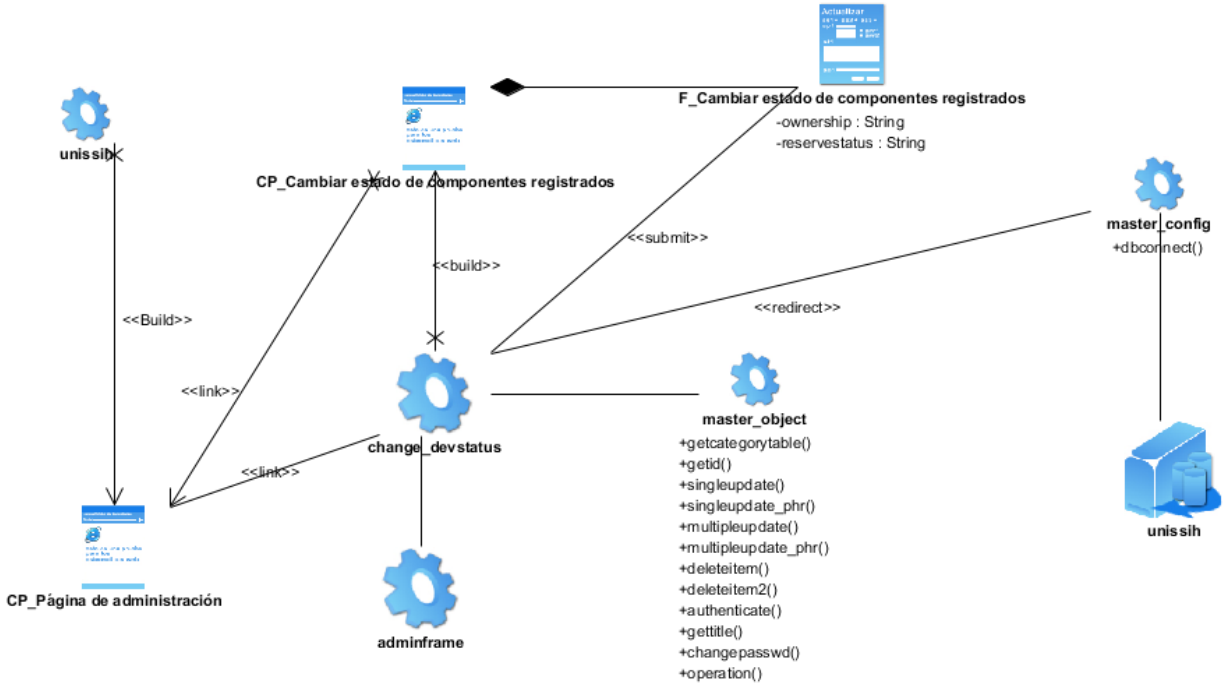

Figura 19: Cambiar estado de los componentes registrados

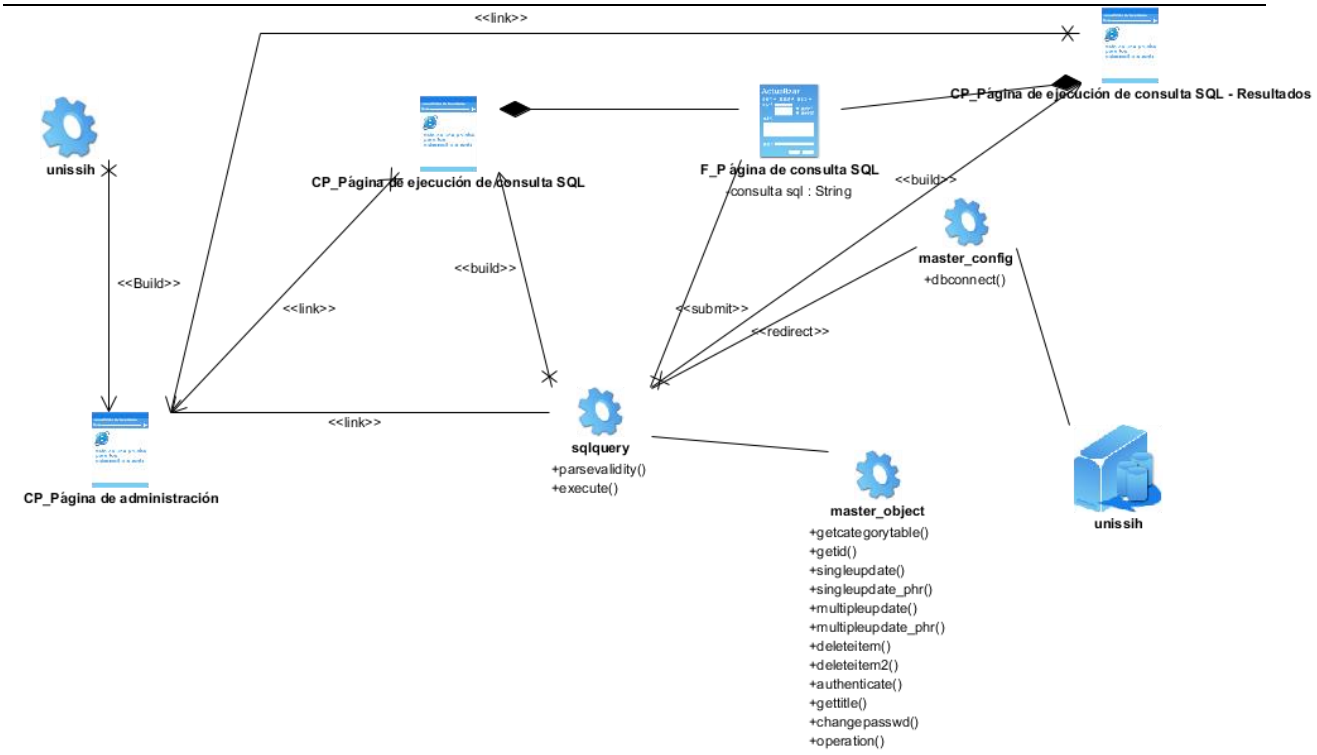

Figura 20: Ejecutar consulta SQL

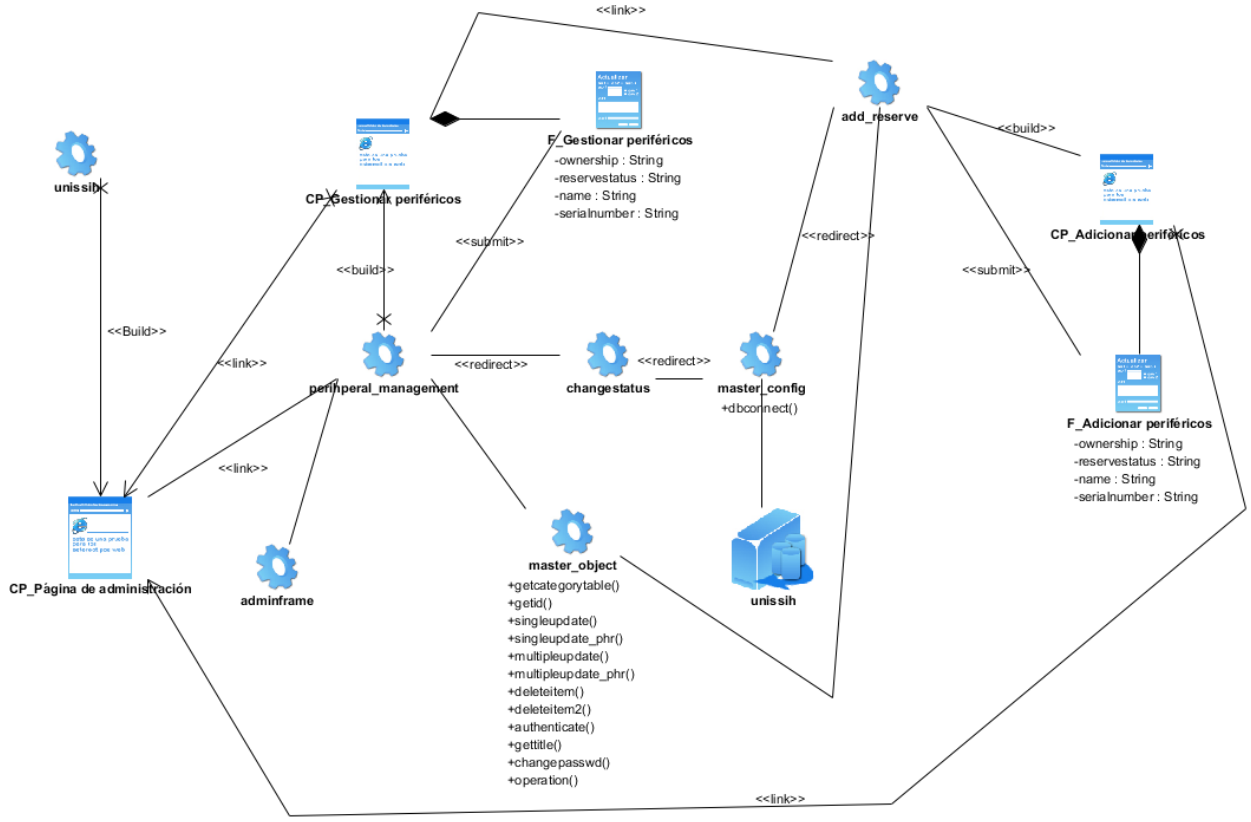

Figura 21: Gestionar periféricos

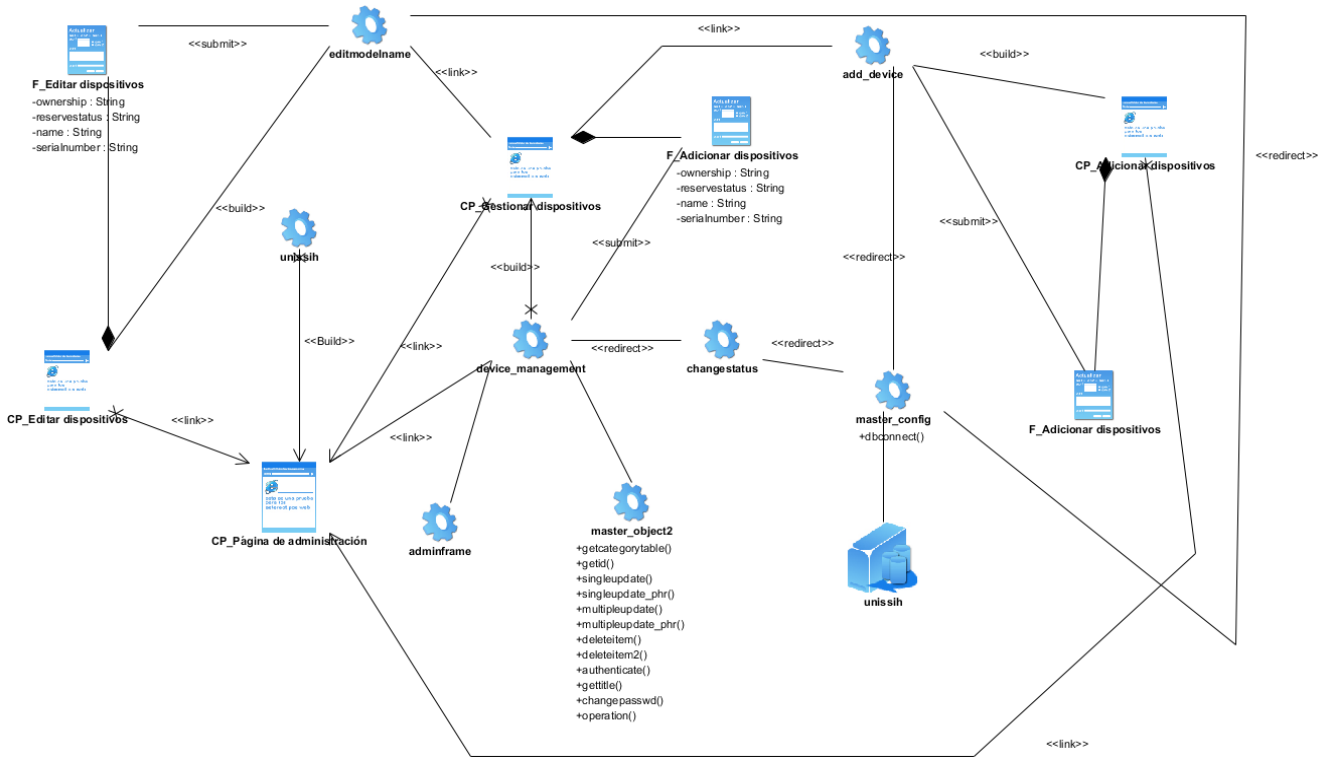

Figura 22: Gestionar dispositivos

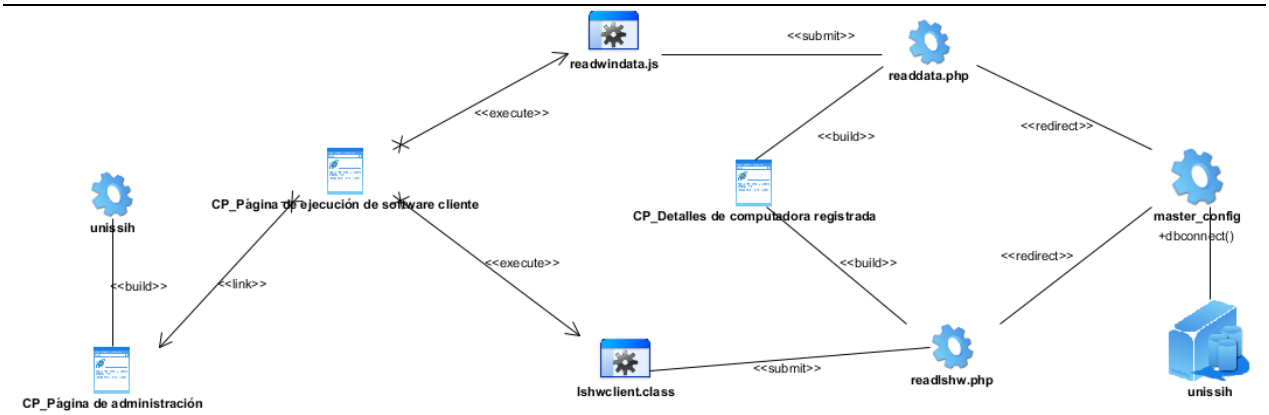

Figura 23: Ejecutar software cliente de extracción de hardware

3.2 Diagrama de clases persistentes

En el diagrama de clases persistentes aparecen las clases que persisten, las cuales poseen la capacidad de mantener su valor en el espacio y en el tiempo. (Rumbaugh, Booch, & Jacobson, 2006). Está compuesto por clases, asociaciones y atributos; interfaces, con sus operaciones y constantes; métodos; información sobre los tipos de atributos, entre otros.

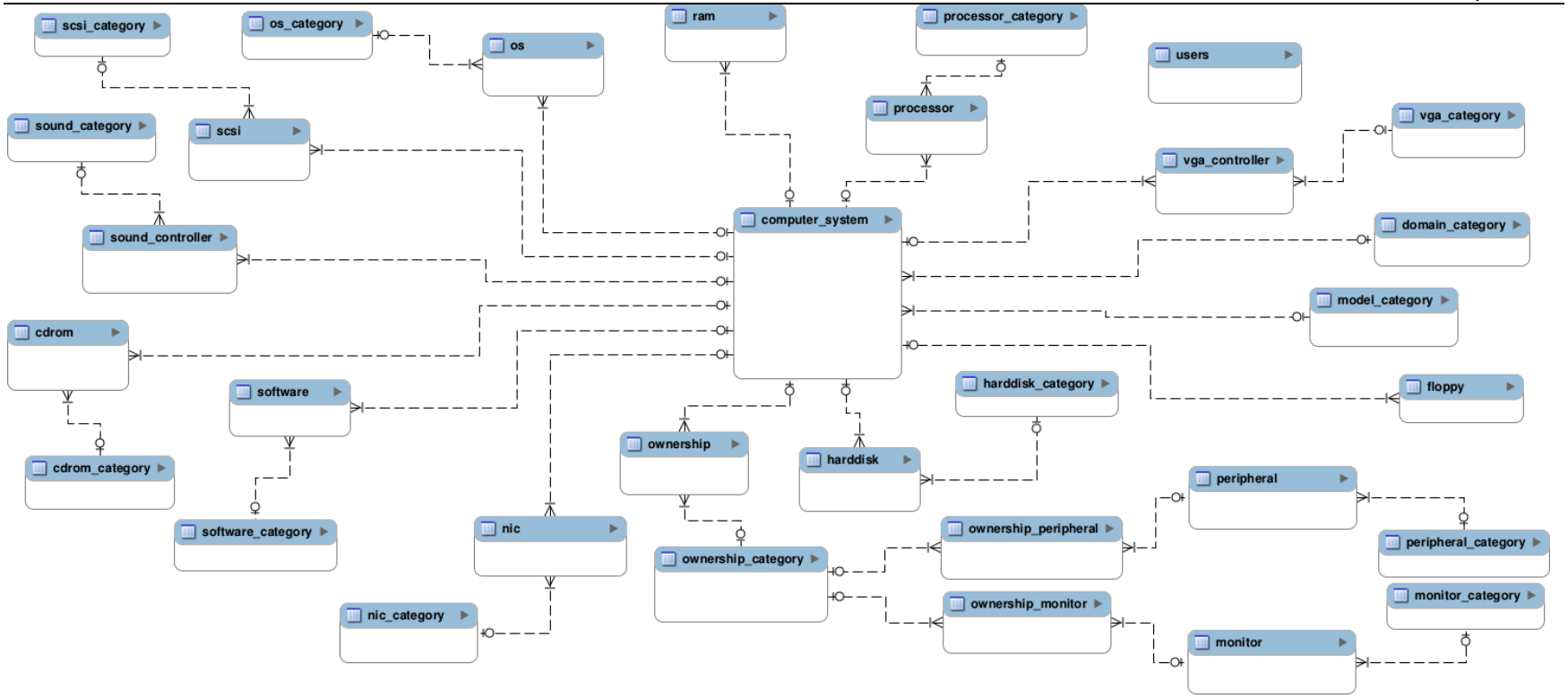

Figura 24: Diagrama de clases persistentes

Modelo de datos

El modelo físico de datos, representa la estructura o descripción física de las tablas de la base de datos, obtenido a partir del modelo lógico de datos. (Rumbaugh, Booch, & Jacobson, 2006)

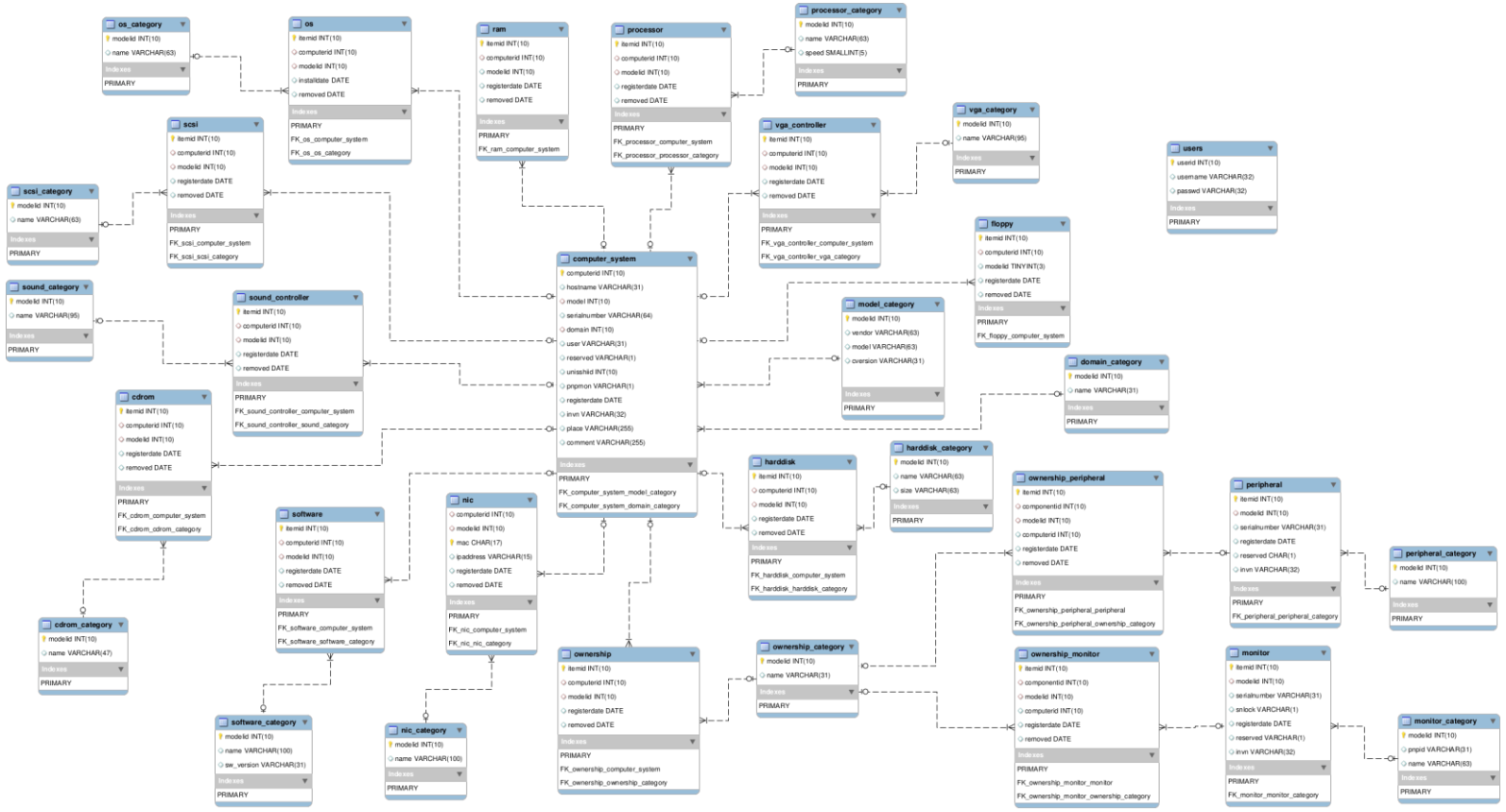

Figura 25: Diagrama de modelo de datos

3.3 Principios de diseño

Diseño de la interfaz del sistema

El software debe brindar una interfaz sencilla que facilite la interacción del usuario con el mismo, para maximizar la productividad y el tiempo de respuesta del usuario ante un requerimiento. La interfaz estará diseñada de modo tal que el usuario pueda tener en todo momento el control de la aplicación, lo que le permitirá ir de un punto a otro dentro de ella con gran facilidad. Se cuidará que la aplicación sea lo más interactiva posible. El ambiente deberá ser con colores claros y fuentes fáciles de leer, sin obstrucción del sistema en lo que realmente le importa al usuario.

Tratamiento de errores

El sistema propuesto valida constantemente la información, con el propósito de minimizar las posibilidades de introducir errores por parte del usuario. En caso de errores se le comunica al usuario a través de un mensaje de alerta en un lenguaje fácil de comprender y se le brinda la oportunidad de intentar la acción nuevamente.

Concepción general de la ayuda

El sistema contará con una ayuda que explicará de manera clara y detallada al usuario todas las funcionalidades del sistema. La ayuda guiará al usuario por todas las funcionalidades, dando especial importancia a las más complejas, con el objetivo de que pueda explotar a fondo toda la usabilidad del sistema, como se espera de un usuario avanzado. La ayuda quedará conformada por una página web con hipervínculos e imágenes que le facilitará al usuario poder ir de un lugar a otro sin perderse. El enlace a la ayuda será visible desde todas las páginas de la aplicación.

Seguridad

El sistema mantiene un fuerte mecanismo de seguridad, basado en un nombre de usuario y contraseña para el acceso al mismo. El jefe de seguridad informática es el encargado de gestionar la información presente en la base de datos del sistema, y cuenta con un nombre de usuario y contraseña únicos, evitando el acceso de usuarios sin autentificar. La información de usuario y contraseña está almacenada en la base de datos, que también provee una capa de seguridad mediante el sistema gestor de base de datos, pero además la contraseña está almacenada de manera encriptada con el algoritmo de hash MD5.

3.4 Estándares de codificación

Actualmente se hallan estándares de codificación para la mayoría de los lenguajes existentes. El uso de ellos partiendo de las convenciones definidas permite una mejor comunicación entre los programadores creando las condiciones para la reusabilidad y el mantenimiento de los sistemas. Por lo que se decide que las variables, nombres de funciones, de consultas y objetos del documento son cortos, claros y describen su propósito. Los nombres de las clases se escriben con mayúscula, las variables con minúsculas y las funciones y métodos que están compuestas por más de una palabra se escribe primero con minúscula y la primera letra de las demás palabras con mayúscula.

Los objetos o tipos de control se nombran según el valor de su contenido. Los inicios y cierre de ámbito se encuentran alineados debajo de la declaración a la que pertenecen y se evitan si hay sólo una instrucción. Los signos lógicos y de operación se separan por un espacio antes y después de los mismos.

3.5 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo del diseño, se implementan en términos de componentes, describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en los lenguajes de programación utilizados y cómo dependen unos componentes de otros. (Jacobson, Booch, & Rumbaugh, 2000)

3.5.1 Diagrama de despliegue

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuyen las funcionalidades entre los nodos de cómputo. (Jacobson, Booch, & Rumbaugh, 2000)

El modelo representa dos nodos: en el lado del servidor se encuentra Apache HTTP Server y MySQL como servicios HTTP y de base de datos respectivamente. Del lado del cliente tenemos el software que hace el inventario de hardware. Se visualiza la aplicación en el lado del cliente utilizando un navegador web conectándose al servidor. Esta comunicación entre cliente y servidor se hace mediante el protocolo TCP/IP.

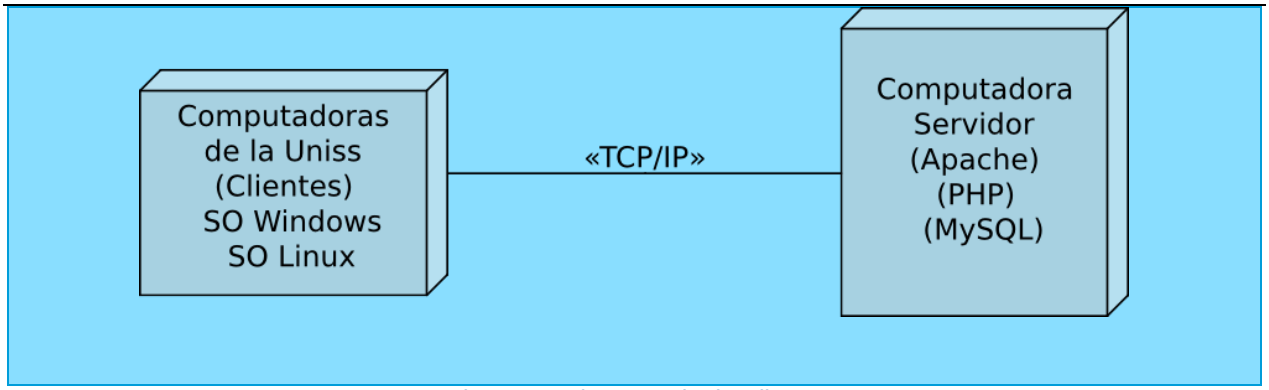

Figura 26: Diagrama de despliegue

3.5.2 Diagrama de componentes

Un diagrama de componentes muestra un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones. Se utiliza para modelar la vista estática de un sistema. Deja ver la organización y las dependencias lógicas entre un conjunto de componentes, sean éstos componentes de código fuente, librerías, binarios o ejecutables. (Rumbaugh, Booch, & Jacobson, 2006)

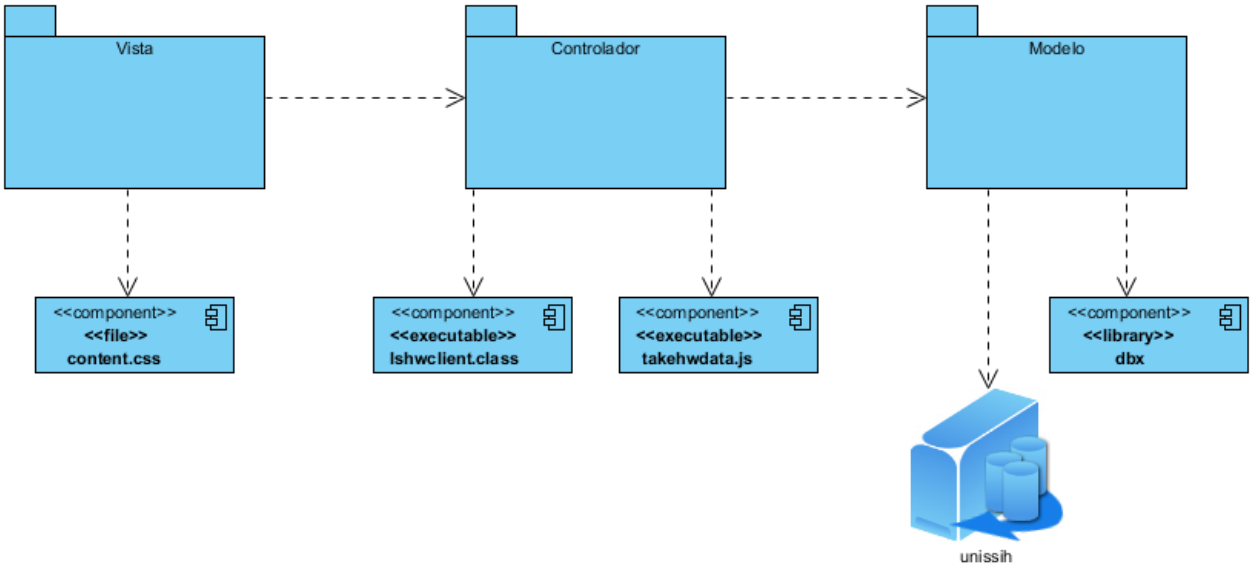

Figura 27: Diagrama de componentes

Componente	Descripción
Vista	Dentro de este subsistema se encuentran todas las clases que muestran una interfaz de usuario.
content.css	Este componente representa el archivo de estilos que usa la aplicación para crear el tema visual en todas las páginas.
Controlador	Este subsistema representa la lógica del negocio, dentro de él se encuentran las clases entidades con las entidades del negocio y sus atributos.
lshwclient.class	Este componente representa el software cliente de Linux, que extrae la información de hardware de las PC con dicho sistema operativo.
takehwdata.js	Este componente representa el software cliente de Microsoft Windows, que extrae la información de hardware de las PC con dicho sistema operativo.
Modelo	Este componente representa la clases que facilitan el acceso a datos a través de la cual se van a comunicar las clases entidades del modelo y la base de datos. Este componente es usado desde el subsistema Controlador.
dbx	Este componente representa un módulo de PHP, que se utiliza para comunicarse con la base de datos. Crea una capa de abstracción que permite utilizar varios SGBD sin migrar todo el código.
unissih	Representa la base de datos de la aplicación.

Tabla 22: Descripción de los componentes

Conclusiones parciales del capítulo 3

Durante el desarrollo de este capítulo se elaboró el diagrama de componentes en el cual se graficó de manera clara las relaciones entre los subsistemas, los componentes y la base de datos; el diagrama de despliegue y los modelos lógicos y físicos de la base de datos. Además se describieron los principios de diseño seguidos, específicamente el diseño de la interfaz de

usuario, los estándares de codificación, la concepción de la ayuda, el tratamiento de excepciones y la seguridad del sistema.

Conclusiones

Con la realización del presente proyecto se arribó a las siguientes conclusiones:

1. El estudio de los fundamentos teóricos y metodológicos para la elaboración de un software que facilite la gestión de inventario de hardware en la Uniss permitió determinar que la metodología RUP es la adecuada para el análisis, diseño, implementación y documentación del sistema, lo cual posibilitó la adecuada documentación de la solución propuesta.

2. Se diseñó un software que facilite la gestión de inventario de hardware en la Uniss que abarca todo el proceso de la gestión sustentado en las bases del software libre, la programación orientada a objetos, el modelo cliente/servidor y la arquitectura en tres capas lo cual le brinda al sistema una mayor flexibilidad y capacidad de mantenimiento.

3. Se implementó un software que soluciona la problemática existente utilizando como lenguaje de programación PHP, Java, JavaScript (JScript), y como sistema gestor de bases de datos MySQL, por ser tecnologías software libre y tener alta portabilidad. Los cuales se integraron correctamente para lograr extraer de manera automática todos los datos necesarios y ayudar a la gestión de los inventarios de hardware en la Uniss.

Recomendaciones

- Permitir a los usuarios sin autentificación el acceso a la información de hardware de su computadora.
- Agregar la posibilidad de que el software cliente consulte automáticamente la información de hardware de los periféricos USB.

Bibliografía

Arteaga Mejía, L. M. (2012). ¿Qué es el Software Libre? GNU Operating System. Recuperado abril 29, 2012, a partir de http://www.gnu.org/philosophy/free-sw.es.html

Bartle, P. (2009). *Información para la gestión y gestión de la información*. Recuperado el 7 de marzo de 2013, de Información para la gestión y gestión de la información: http://www.scn.org/mpfc/modules/mon-miss.htm

Belmonte, O. (2009). *Introducción al lenguaje de Java. Una guía básica.*

Burbeck, S. (1992). *Applications Programming in Smalltalk*-80(TM): How to use Model-View-Controller (MVC). Recuperado a partir de http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

Calderón. (2009). *Metodologías ágiles.*

Curto, J. (2006). *Information Management*. Recuperado el 6 de marzo de 2013, de Reflexiones sobre las tecnologías de la información: http://informationmanagement.wordpress.com

Dante, G. P. (1998). Principios, conceptos y aplicaciones. En G. P. Dante, *Gestión de información en las organizaciones.* Santiago de Chile: Universidad de Chile.

Date, C. J. (2001). *Introducción a los sistemas de bases de datos.*

Elizalde, R. (2004). *¿Qué pasa con Cuba?* Recuperado el 28 de febrero de 2013, de http://www.cubavision.cubaweb.cu/prensa_detalles

Free Software Foundation. Recuperado el 21 de mayo del 2014 de http://www.fsf.org/

Free Software Foundation (2009) La Definición de Software Libre. Recuperado de http://www.gnu.org/philosophy/free-sw.es.html

González, A. (2005). *Modelamiento del negocio. Centro de Estudios de Ingeniería de Sistemas* (CEIS).

Hernández M., Carlos C., Cespón, R. (1998). *Logística de la distribución comercial, un enfoque sistémico*. Recuperado de Revista Logística Aplicada. Ciudad de la Habana.

Hernández, E. (2010). *Lenguaje de Modelado.* Recuperado el 22 de enero de 2013, de http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

Introducción a la Programación Orientada a Objetos. (2004). La Habana: Editorial Felix Varela.

Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *El proceso unificado de desarrollo*. Madrid: Addison Wesley.

Jacobson, I., Booch, G., & Rumbaugh, J. (2006). *El Proceso Unificado de Desarrollo de Software.* La Habana: Félix Varela.

La definición de Open Source. Recuperado de *http://www.opensource.org/docs/definition.php*

Leal, E. T. (2008). Las tecnologías de la información y las comunicaciones (TIC) y la brecha digital. *Revista de Universidad y Sociedad del Conocimiento*, 3-6.

Manage. (2007). Free Download Manager: *Paradigma Visual para UML*. Recuperado el 5 de marzo de 2007, de http://www.freedownloadmanager.org/es/downloads/contact.php.

Marcial, A. (1996). Información: Una nueva propuesta conceptual. En A. Marcial, *Información: Una nueva propuesta conceptual* (pág. 190).

Mateu, C. (2004) *Desarrollo de aplicaciones web*. España.

Mato García, R. M. (2006). Sistema de Base de Datos. La Habana: Félix Varela.

Morales Flores, E. (2 de noviembre de 2007). *La gestión y los gestores de la información.* Recuperado el 20 de febrero de 2013, de Bibliodocencia: http://www.bibliodocencia.com

Morero, F. (2000). *Introducción a la OOP*. Madrid: Grupo EIDOS.

MUSCIANO, C., KENNEDY, B. HTML, La Guía Completa. México, D.F.: The McCraw-Hill Companies, Inc, 1999. ISBN 970-10-2141-X.

Negrino, T y Smith, D. (2000) *Guía de aprendizaje JavaScript*. Tercera edición. Madrid: PRENTICE HALL.

NetBeans. (2011). NetBeans IDE. org : IDE de Desarrollo. Recuperado de https://netbeans.org/features/index.html.

Palacios, E. (2008). *Aplicaciones ricas en Internet (RIA). Un enfoque de refactorización.*

Popkin Software and System. *(2005). Recuperado el 10 de mayo de 2011, de Modelado de sistemas con UML: http://es.tldp.org/tutoriales/doc-modelado-sistemas-UML-multiple-html/c124.html*

Potencier, F y Zaninotto, F. (2008) *Symfony, la guía definitiva*. S.l.

Pressman, R. S. (2007). *Ingeniería del Software Un enfoque práctico.*La Habana: Félix Varela.

Risco Nuñez, A. (2011). Sistema de Gestión de Base de Datos. Conferencia presentado en Base de Datos, Universidad de Sevilla, Sevilla, España.

Riveros, F. (2008). *Gestor de Base de Datos: MySQL, PostgreSQL, SQLite*. Recuperado el 6 de marzo de 2013, de http://www.eaprende.com/base_de_datos_SQL_Server_con_PHP_y_ADODB.html

Sæther, B., Stig, A., Alexander, S., Egon, W., Jim, T., Lars, L., Rasmus, S., Zeev, Z. (2001) *Manual de PHP*. S.l.

Sandoval, L. (21 de Septiembre de 2009). *Gestores de Bases de Datos.* Recuperado el 25 de marzo de 2013, de http://www.chacharaselnido.com/ITVG/GBD%20EXPO.pptx.

SELECTING A DEVELOPMENT APPROACH. Revalidated: March 27, 2008. Retrieved 27 Oct 2008. https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf

Sistemas, Centro de Estudios y Diseño de. 2005. CEDS e- Learnind: Introducción a los Sistemas y Herramientas CASE. [En línea] Registro Gral. de la Propiedad Intelectual de Madrid, 2005. [Citado el: 12 de abril de 2013.] http://ceds.nauta.es/informes/case01.htm.

Stallman, R. Software libre para una sociedad libre. Recuperado de http://biblioweb.sindominio.net/pensamiento/softlibre/softlibre005.html#toc28

The OSI Approved Licenses. Recuperado de http://www.opensource.org/licenses/index.php

The PostgreSQL Global Development Group. (2003). PL/pgSQL - SQL Procedural Language. En T. G. Development Group, *PostgreSQL 7.4devel Documentation.* The PostgreSQL Global Development Group.

Anexos

Anexo 1 Ficha técnica

PC:
Fecha:
Operador Principal:

No.	AGREGADO	MARCA	No. Serie/Identificación
1.	Chasis		
2.	Monitor		
3.	Teclado		
4.	Mouse		
5.	Bocinas		
6.	Lector CD-ROM		
7.	Disco Duro		
8.	Adaptador de Vídeo		
9.	Mother Board		
10.	Tarjeta Red		
11.	Tarjeta Sonido		
12.	Sistema Operativo		
13.	UPS		
14.	Procesador		
15.	Memoria RAM		
16.	Fuente Interna		
17.	Impresora		

Programas Instalados

Figura 28: Ficha técnica

Anexo 2 Guía de la entrevista

Entrevista realizada a directivos de la Uniss para conocer sobre el desarrollo de la gestión de inventario de hardware en la Uniss.

1. ¿Cuándo llegaron las primeras computadoras a la universidad?
2. ¿Qué características tenían?
3. ¿Quiénes eran los encargados?
4. ¿Cómo se realizaba la gestión de inventario de hardware?
5. ¿Con que frecuencia se realiza este proceso?

Anexo 3 Prototipo de Interfaz. Caso de uso: Autentificar Usuario

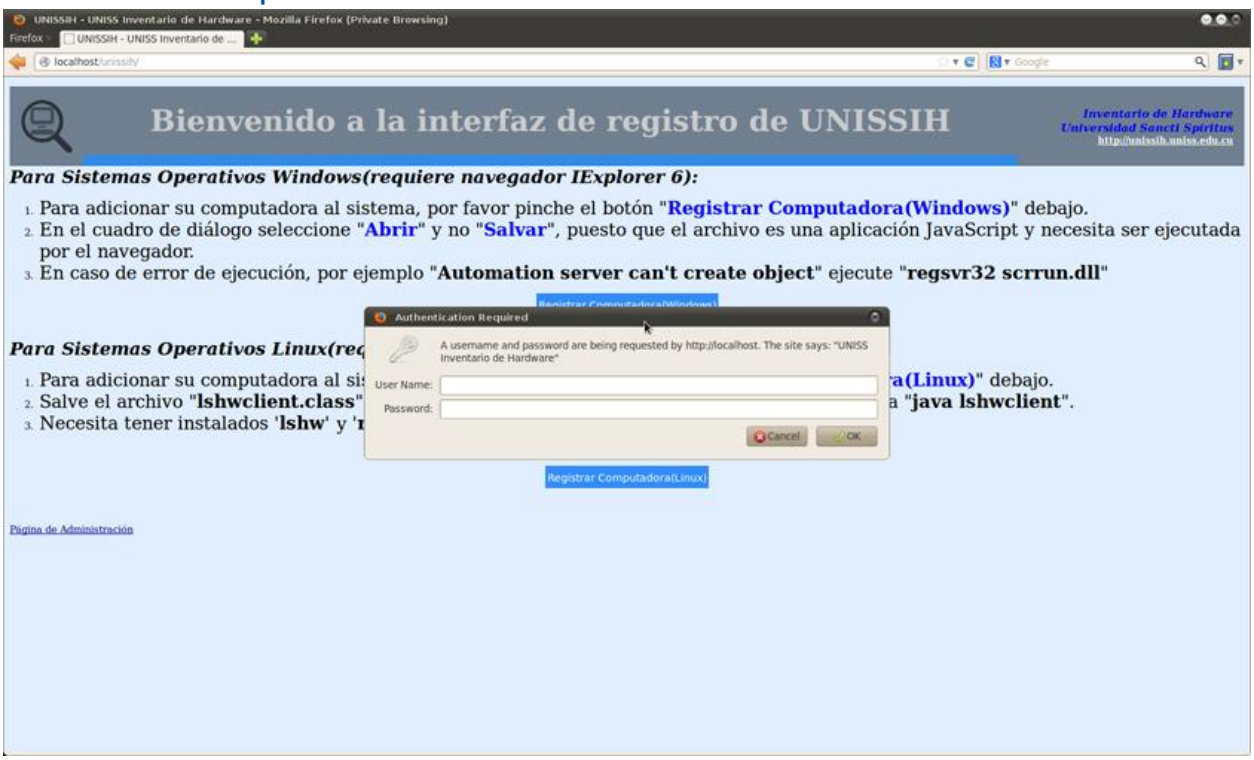

Figura 29: Prototipo de Interfaz. Caso de uso: Autentificar usuario

Anexo 4 Prototipo de Interfaz. Caso de uso: Cambiar Contraseña

Figura 30: Prototipo de Interfaz. Caso de uso: Cambiar contraseña

Anexo 5 Prototipo de Interfaz. Caso de uso: Mostrar computadoras registradas

Figura 31: Prototipo de Interfaz. Caso de uso: Mostrar computadoras registradas

Figura 32: Prototipo de Interfaz. Caso de uso: Mostrar computadoras registradas

Anexo 6 Prototipo de Interfaz. Caso de uso: Mostrar categoría de componentes

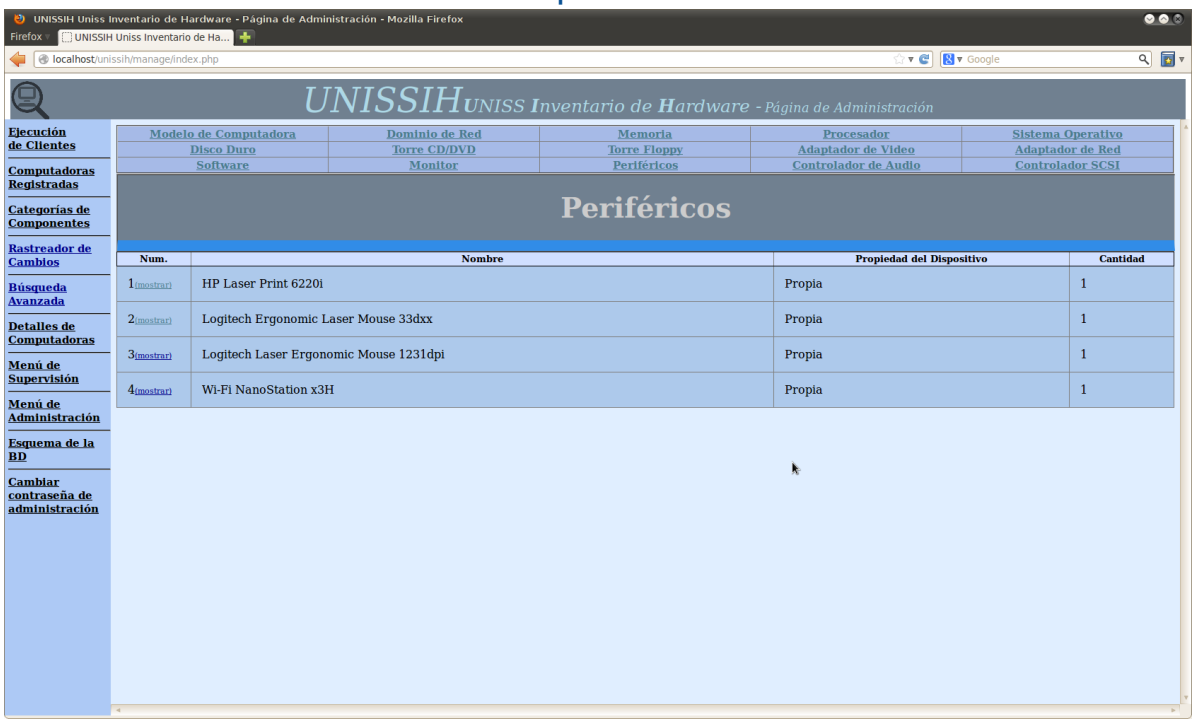

Figura 33: Prototipo de Interfaz. Caso de uso: Mostrar categoría de componentes

Figura 34: Prototipo de Interfaz. Caso de uso: Mostrar categoría de componentes

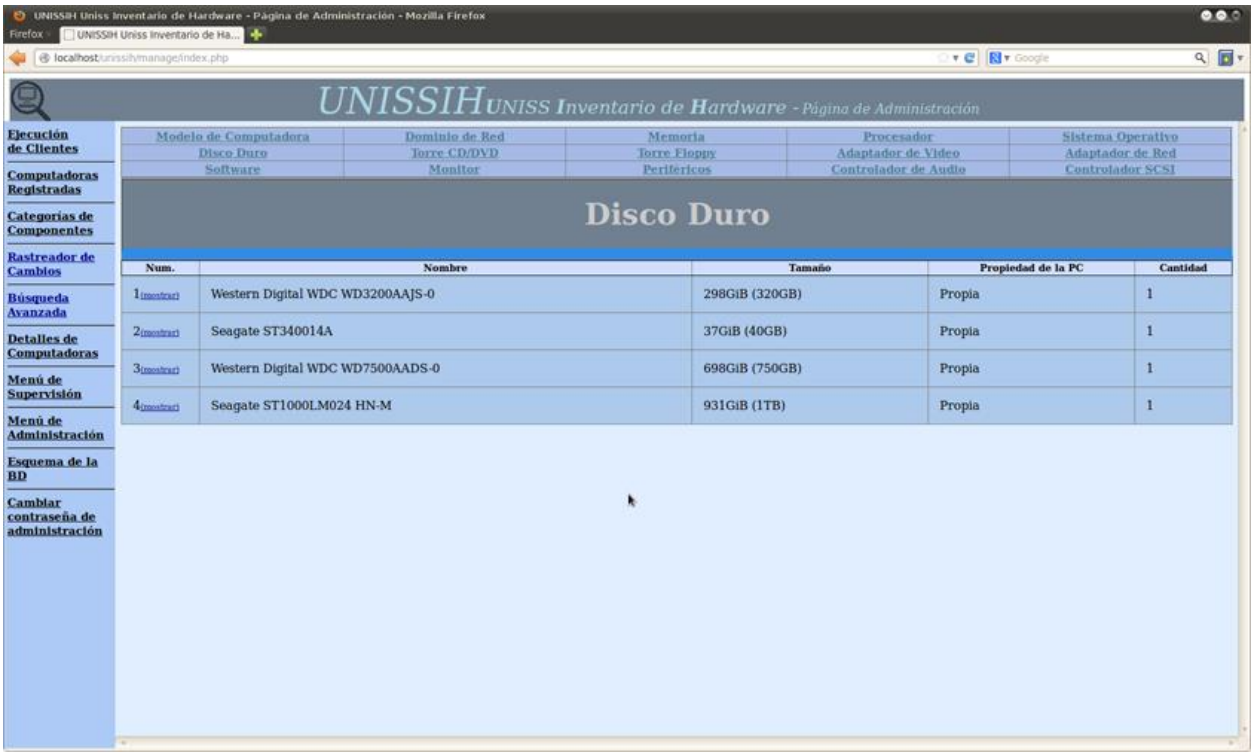

Figura 35: Prototipo de Interfaz. Caso de uso: Mostrar categoría de componentes

Anexo 7 Prototipo de Interfaz. Caso de uso: Rastrear cambios de hardware

Figura 36: Prototipo de Interfaz. Caso de uso: Rastrear cambios de hardware

Figura 37: Prototipo de Interfaz. Caso de uso: Rastrear cambios de hardware

Figura 38: Prototipo de Interfaz. Caso de uso: Rastrear cambios de hardware

Figura 39: Prototipo de Interfaz. Caso de uso: Rastrear cambios de hardware

Figura 40: Prototipo de Interfaz. Caso de uso: Rastrear cambios de hardware

Anexo 8 Prototipo de Interfaz. Caso de uso: Mostrar historial de cambios

Figura 41: Prototipo de Interfaz. Caso de uso: Mostrar historial de cambios

Figura 42: Prototipo de Interfaz. Caso de uso: Mostrar historial de cambios

Figura 43: Prototipo de Interfaz. Caso de uso: Mostrar historial de cambios

Anexo 9 Prototipo de Interfaz. Caso de uso: Búsqueda avanzada

Figura 44: Prototipo de Interfaz. Caso de uso: Búsqueda avanzada

Figura 45: Prototipo de Interfaz. Caso de uso: Búsqueda avanzada

Figura 46: Prototipo de Interfaz. Caso de uso: Búsqueda avanzada

Anexo 10 Prototipo de Interfaz. Caso de uso: Mostrar estructura de la base de datos.

Figura 47: Prototipo de Interfaz. Caso de uso: Mostrar estructura de la base de datos

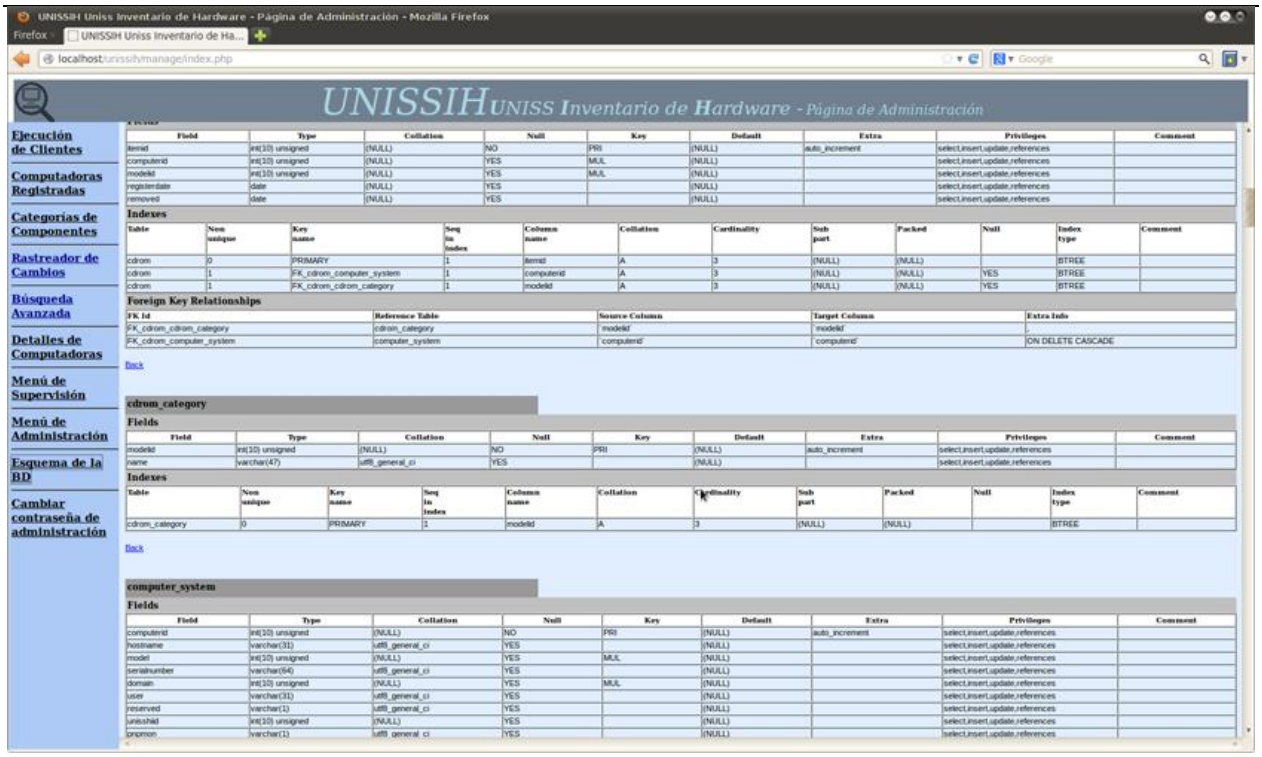

Figura 48: Prototipo de Interfaz. Caso de uso: Mostrar estructura de la base de datos.

Anexo 11 Prototipo de Interfaz. Caso de uso: Mostrar software instalado

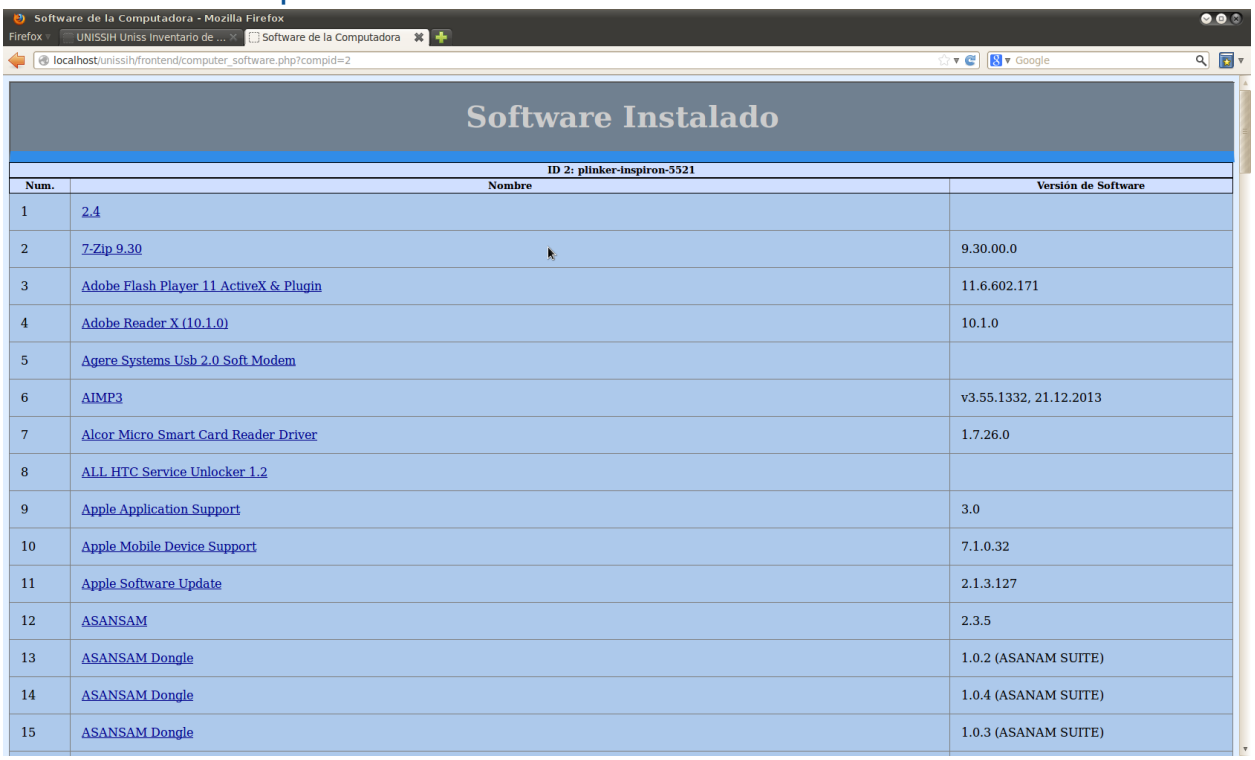

Figura 49: Prototipo de Interfaz. Caso de uso: Mostrar software instalado

Anexo 12 Prototipo de Interfaz. Caso de uso: Gestionar detalles de computadoras

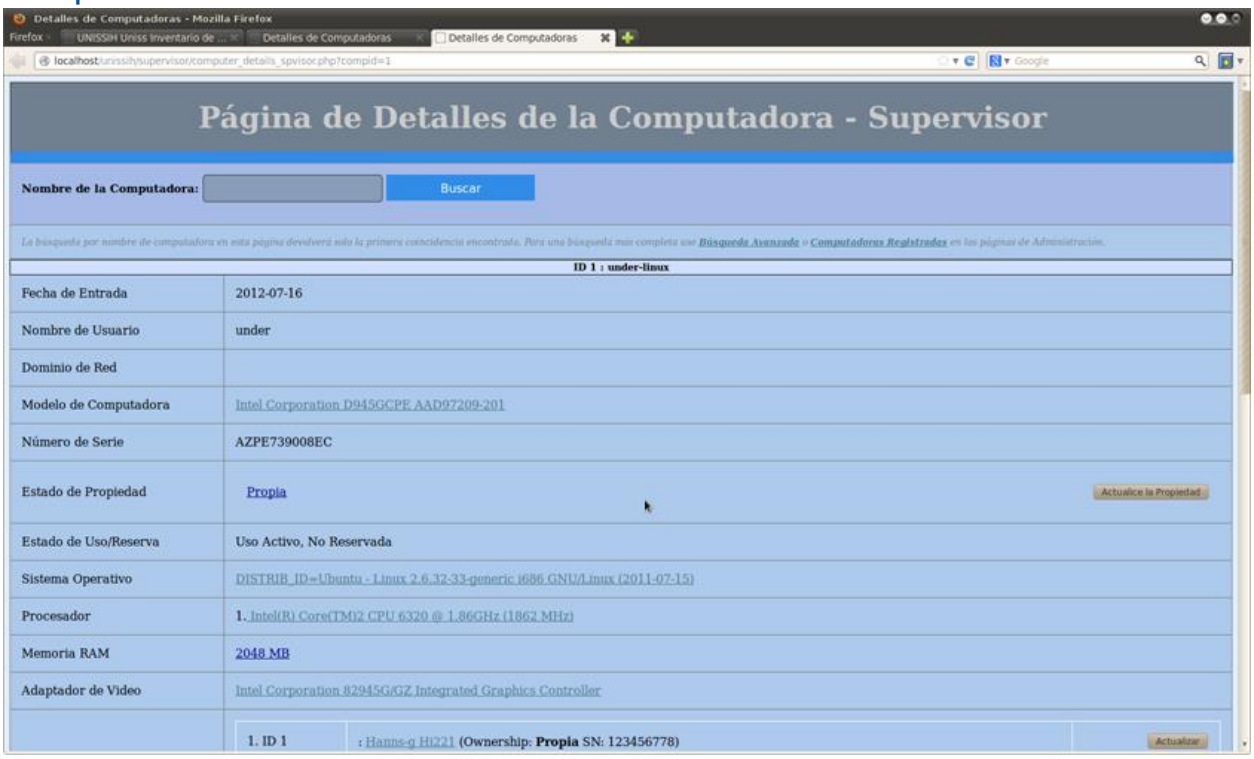

Figura 50: Prototipo de Interfaz. Caso de uso: Gestionar detalles de computadoras

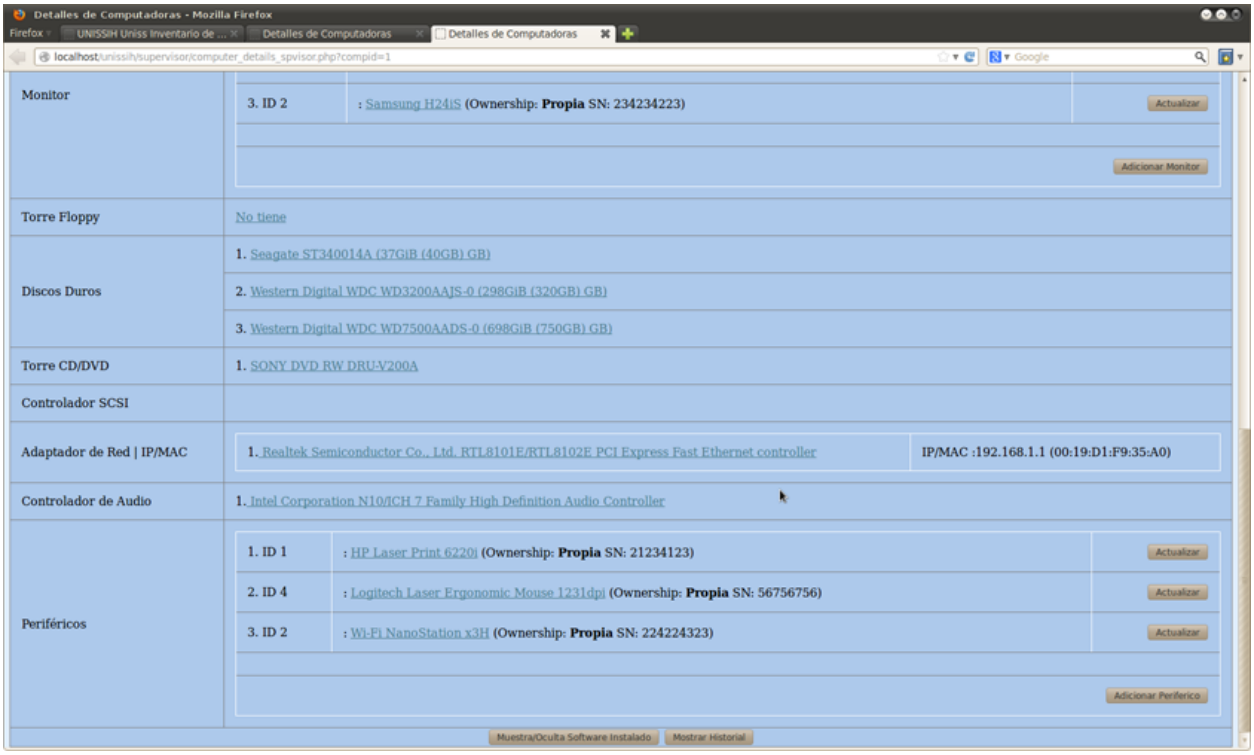

Figura 51: Prototipo de Interfaz. Caso de uso: Gestionar detalles de computadoras

Anexo 13 Prototipo de Interfaz. Caso de uso: Cambiar estado de las computadoras registradas

Figura 52: Prototipo de Interfaz. Caso de uso: Cambiar estado de las computadoras registradas

Anexo 14 Prototipo de Interfaz. Caso de uso: Cambiar estado de los componentes registrados

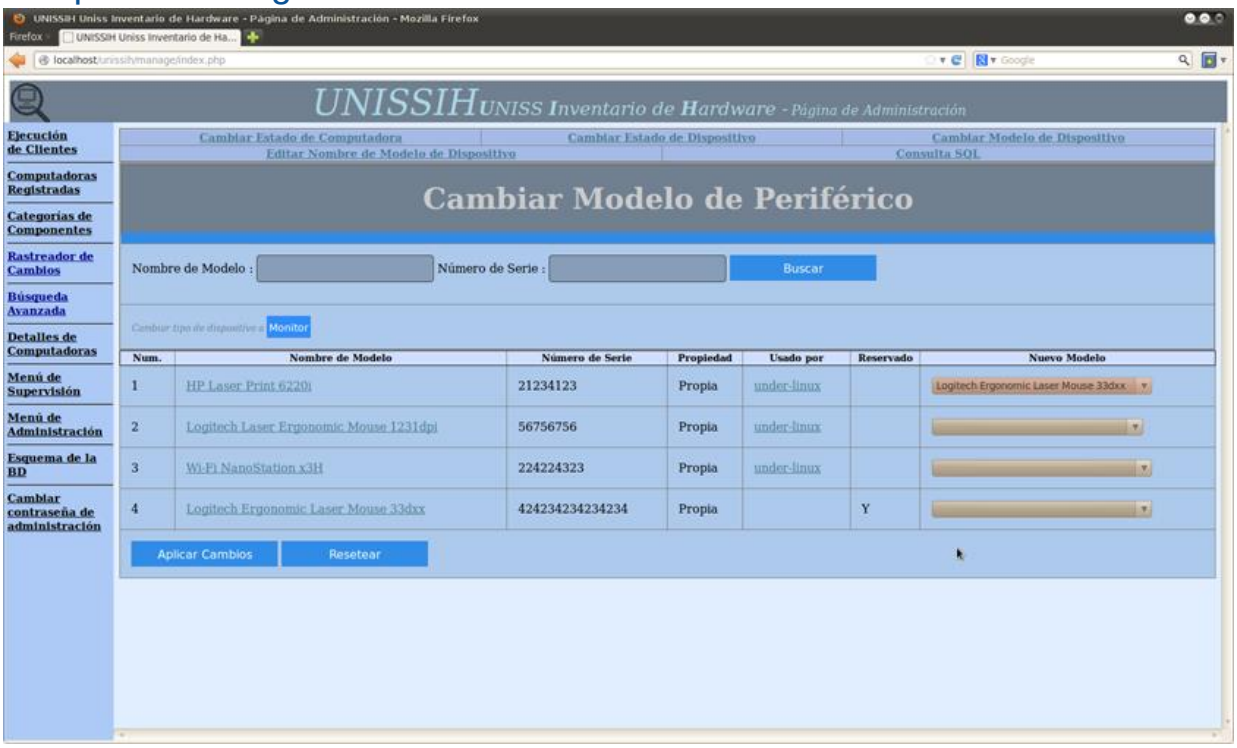

Figura 53: Prototipo de Interfaz. Caso de uso: Cambiar estado de los componentes registrados

Anexo 15 Prototipo de Interfaz. Caso de uso: Ejecutar consulta SQL

Figura 54: Prototipo de Interfaz. Caso de uso: Ejecutar consulta SQL

Anexo 16 Prototipo de Interfaz. Caso de uso: Gestionar periféricos

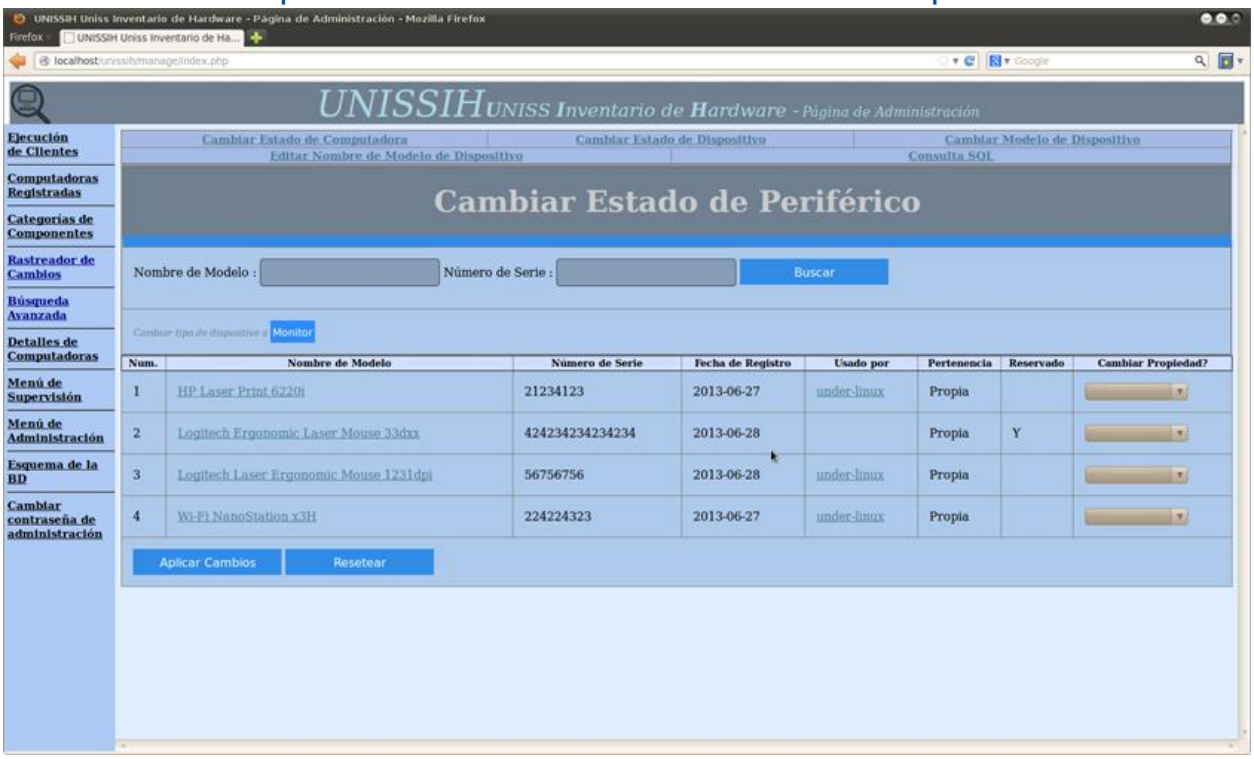

Figura 55: Prototipo de Interfaz. Caso de uso: Gestionar periféricos

Anexo 17 Prototipo de Interfaz. Caso de uso: Gestionar dispositivos

Figura 56: Prototipo de Interfaz. Caso de uso: Gestionar dispositivos

Anexo 18 Prototipo de Interfaz. Caso de uso: Ejecutar software cliente de extracción de hardware

Figura 57: Prototipo de Interfaz. Caso de uso: Ejecutar software