Universidad de las Ciencias Informáticas

Facultad 1

Sistema de Certificación y Homologación de Hardware para Nova GNU/Linux

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

Autor: Ricardo Ramón Carbajo Pérez

Tutores: Ing. Yordanis Cabreja Nuñez

Ing. Eduardo Alejandro Cuesta Llanes

La Habana, Junio 2012

"Año 54 de la Revolución"

Declaración de autoría

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

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

_____

Ricardo Ramón Carbajo Pérez

_____ _____

Ing. Yordanis Cabreja Nuñez Ing. Eduardo Alejandro Cuesta Llanes

Agradecimientos

A nuestro Comandante en Jefe y a la Revolución por permitirme cumplir este sueño.

A mis queridos padres, por su guía, confianza y apoyo constante.

A todos mis profesores que han contribuido a mi formación durante estos años.

A todos los que de una u otra forma han colaborado en la realización de este trabajo.

A todos muchas gracias.

Dedicatoria

A mis padres, que han sido la voluntad, la perseverancia y la guía de todos estos años.

A todos mis seres queridos y la universidad que tanto aportó en mi formación.

Resumen

El presente trabajo de diploma se centra en la necesidad de proveer a la Plataforma de Migración a Software Libre un sistema de certificación y homologación de hardware para Nova, que permita gestionar la validación de la información de compatibilidad del hardware que se encuentra en las instituciones cubanas. El cual es necesario en el proceso de migración a Software Libre que se lleva a cabo en todo el país, debido a la gran variedad de hardware y a la incompatibilidad que estos han presentado durante la migración. Se realiza un análisis de los principales sistemas de certificación de hardware, así como los principales factores que provocan la incompatibilidad del hardware con los sistemas operativos libres. Mediante una caracterización se muestran los lenguajes, herramientas y tecnologías que se emplean en la construcción del sistema, entre los que se encuentran como lenguaje PHP, el framework Symfony, la librería jQuery y el sistema gestor de bases de datos PostgreSQL, como soporte al desarrollo de la solución propuesta. Se identifican, prueban y documentan todas las funcionalidades que el sistema debe cumplir, donde el proceso de desarrollo es guiado por la metodología ágil SXP. El sistema propuesto cumple con los requisitos establecidos por el cliente, permitiendo gestionar la información referente a la compatibilidad del hardware, integrado a la Plataforma de Migración mediante un conjunto de servicios web.

Palabras Clave: certificación, compatibilidad, hardware, homologación, migración

Índice general

Índice de figuras

Índice de tablas

Introducción

El avance de las tecnologías ha acelerado el desarrollo de nuestra sociedad, provocando cada vez más que el uso de estas sea indispensable por la complejidad de los procesos socio-económicos. Un elemento importante en el desarrollo de cualquier país es la información, su gestión es necesaria y las Tecnologías de la Información y las Comunicaciones (TIC) desempeñan un papel fundamental permitiendo la rápida comunicación e intercambio de información [15].

En contraste al desarrollo alcanzado, la globalización y el consumismo han desencadenado tendencias económicas corrosivas motivadas por el incremento del capital económico, los intereses de las compañías y las ansias de poder. Uno de los fenómenos que más afecta el desarrollo es la brecha digital [23], que separa los países ricos de los pobres, obligando a los últimos a consumir las tecnologías de los países desarrollados. El uso del software como complemento para el desarrollo es una necesidad, pero para países pobres del tercer mundo el costo de implantación de software privativo puede llegar a tener precios exorbitantes, y más aún a largo plazo.

En contraposición al software privativo surge el Software Libre (SWL) como una filosofía que defiende las libertades para la libre cooperación y promoviendo un marco de trabajo colaborativo. Por sus ventajas socio-económicas muchos países y organizaciones se han trazado como objetivo hacer una paulatina transición a plataformas libres. Cuba es uno de esos países que se ha propuesto la migración a SWL, pues el bloqueo económico, financiero y comercial instrumentado por el gobierno norteamericano contra Cuba en una de sus cláusulas prohíbe el uso y/o adquisición de los sistemas informáticos producidos por firmas norteamericanas, por empresas y/o usuarios cubanos. Teniendo en cuenta que el 80 % de software en Cuba proviene de empresas norteamericanas resulta perjudicial mantener su uso[26]. Además, el software privativo en Cuba no es viable, debido a este aumenta sus requerimientos tecnológicos continuamente y la mayor parte del parque tecnológico en el país tiene un gran periodo de desactualización. Y económicamente no puede sostenerse por concepto de pago de licencias, pues su actualización requiere invertir una gran suma de capital en nuevas tecnologías.

El SWL es la alternativa para lograr la soberanía tecnológica pues le permite a Cuba mayor independencia de tecnologías foráneas y brinda mayor seguridad nacional [14] respecto al software privativo, debido que es un software auditado y probado en comunidades de todo el mundo, que evoluciona a la par de las nuevas tecnologías. Si se implementa en Cuba

permitiría ser creado por y para los cubanos logrando mayor socio-adaptabilidad, así como un óptimo aprovechamiento de la infraestructura tecnológica del país. Además de que una de las premisas de la Revolución es la socialización del conocimiento, la cual está implícita en los objetivos de la filosofía del SWL.

En materia de migración a SWL, en Cuba se han ido dando pasos como lo son la creación del Grupo Nacional para la migración a plataformas de Código Abierto, la Universidad de las Ciencias Informáticas (UCI) que promueve el desarrollo de SWL y la creación de Nova GNU/Linux (Nova) como sistema operativo base para la migración. Además, se implementaron varios pilotos de migración en algunos ministerios del país, donde el proyecto Servicios Integrales de Migración y Soporte (SIMAYS) de la Facultad 1 de la UCI, llevó a cabo la asesoría e impartió cursos de superación sobre herramientas libres. Uno de los factores que influyeron negativamente, fue la incompatibilidad del hardware con Nova, debido a la gran variedad que existe. Al no poder comercializar Cuba directamente con los fabricantes de hardware por el bloqueo económico, está obligada a hacer compras mediante varios proveedores en todo el mundo, creando así un parque tecnológico muy extenso y heterogéneo.

A partir de la experiencia adquirida por SIMAYS en la migración de varias empresas, surgió la idea de una Plataforma para la Migración a Software Libre (PMSWL) que complementara y acelerara la migración de las instituciones. Donde guiada por la Metodología Cubana para la Migración a Software Libre y Código Abierto [26], automatiza de forma centralizada el proceso de migración. Un aspecto definitivo en dicho proceso es la certificación de compatibilidad de hardware con el sistema operativo, que es analizado mediante especialistas en migración a través de la herramienta GITI. La cual no es capaz de almacenar la información de compatibilidad para su posterior análisis, lo que se traduce en un proceso iterativo de análisis del hardware y gasto adicional de tiempo y recursos para validar la compatibilidad. Poder determinar en un espacio reducido de tiempo el hardware compatible, es necesario a la hora de tomar decisiones para no desaprovechar recursos. El hardware como base tecnológica de una institución, puede frenar la migración si es incompatible con el sistema operativo, por lo que en caso de alguna incompatibilidad buscar soluciones de controladores compatibles o hardware homólogo es una necesidad imperante.

Dada la situación problémica anteriormente expuesta se define como **problema científico**: *¿Cómo certificar y homologar hardware para Nova desde la Plataforma de Migración a Software Libre?*

En la presente investigación se tiene como **objeto de estudio**: los sistemas de certificación y homologación de hardware; siendo el **campo de acción**: los sistemas de certificación y homologación de hardware para Software Libre.

Para el desarrollo de la investigación se trazó como **objetivo general**:

Desarrollar un sistema de certificación y homologación de hardware para Nova integrado a la Plataforma de Migración a Software Libre. Del mismo, se pueden derivar los siguientes **objetivos específicos**:

1 Determinar los referentes teóricos de los sistemas existentes que permiten la certificación de hardware.

2 Definir los requisitos del sistema de certificación.

3 Diseñar el sistema.

4 Implementar el sistema diseñado.

5 Probar el sistema de certificación mediante los casos de pruebas.

Se parte de la siguiente **idea a defender**: *El desarrollo del sistema informático permitirá la certificación y homologación de hardware para Nova desde la Plataforma de Migración a Software Libre.*

Para dar cumplimiento a los objetivos de la investigación se definieron una serie de **tareas a desarrollar**, tales como:

1.1 Análisis los sistemas similares de las principales organizaciones a nivel mundial.

2.1 Identificación de los requisitos funcionales y no funcionales que el sistema debe cumplir.

3.1 Definición de una arquitectura candidata.

4.1 Implementación del sistema propuesto en términos de componentes de implementación.

4.2 Implementación de las funcionalidades como servicios para lograr la integración del sistema con la Plataforma de Migración.

5.1 Prueba de los componentes desarrollados independientemente como unidades para validar las funcionalidades.

Para el desarrollo de la presente investigación se usaron métodos empíricos y teóricos. A continuación se definen cada uno de estos.

Entre los **métodos teóricos** se encuentran:

Analítico-Sintético:

Permite la división mental del fenómeno en sus múltiples relaciones y la unión entre las partes previamente analizadas.

Posibilitó descubrir las características generales de los sistemas de homologación y certificación existentes para así crear las ideas concretas que permiten definir la propuesta de esta solución.

Histórico-Lógico:

Analiza la trayectoria completa del fenómeno, su condicionamiento a los diferentes periodos de la historia, revela las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales.

Permitió conocer el proceso de desarrollo y la interacción del hardware y el software desde su surgimiento así como las tendencias actuales y los orígenes de los procesos de certificación y homologación.

Entre los **métodos empíricos** se encuentra:

Observación:

Percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno.

El estudio de los diferentes sistemas que se analizan, permitió determinar las principales funcionalidades necesarias en el sistema que se desea desarrollar y las posibles mejoras a realizarse.

El presente trabajo de diploma consta de 3 capítulos bien estructurados y de forma organizada para un buen entendimiento del lector.

Capítulo 1 Análisis de los diferentes sistemas que a nivel mundial se usan para la certificación de hardware así como sus principales características. Descripción y caracterización de la metodología, las herramientas y tecnologías empleadas en el desarrollo del sistema.

Capítulo 2 Diseño de la solución propuesta guiado por la metodología de desarrollo definida, identificación de funcionalidades y descripción de historias de usuario.

Capítulo 3 Definición de la planificación de los releases, descripción de los casos de pruebas y análisis del impacto de la solución propuesta.

1

Fundamentación teórica de la investigación

1.1 Introducción

En el presente capítulo son analizados los fundamentos teóricos generales de los sistemas de certificación de hardware como modelo de referencia para el sistema propuesto y los servicios web para la integración con la plataforma de migración. Se hace un análisis de las tendencias actuales del hardware y se esclarecen conceptos referentes al software libre, la certificación y la homologación de hardware. Además, se muestran las diferentes herramientas, tecnologías y metodología utilizadas para la implementación del sistema.

1.2 Conceptos generales

1.2.1 Certificación y homologación de hardware

La certificación de hardware tiene vital importancia en el proceso de migración a SWL de una empresa, pues a partir de una extensa base de datos de certificaciones es posible determinar el grado de compatibilidad del hardware con el sistema operativo base seleccionado para la migración. Por tanto, saber qué cantidad de hardware es compatible posibilita tomar decisiones a la hora de seleccionar uno nuevo para la institución y si existe hardware no certificado buscar alternativas o solicitar que se certifique. En ocasiones cuando es necesario un hardware específico que no es compatible con el sistema operativo que disponemos o es difícil su adquisición, se busca uno con características similares u homólogo.

Homologación es: equiparar, poner en igualdad de condición dos cosas. Registrar y confirmar el resultado de una prueba deportiva realizada con arreglo a ciertas normas. Contrastar el cumplimiento de determinadas especificaciones o

características de un objeto o de una acción [17].

La **homologación de hardware** es el procedimiento que prueba la equivalencia entre hardware del mismo tipo pero de diferentes fabricantes y modelos que reúnen características similares y son compatibles con un software de sistema determinado e interactúa de forma correcta con este [12].

El término **certificación** está definido como: documento en que se asegura la verdad de un hecho. Asegurar, afirmar o dar por cierto algo. Obtener, mediante pago, un certificado o resguardo por el cual el servicio de correos se obliga a hacer llegar a su destino una carta o un paquete que se ha de remitir por esa vía. Hacer constar por escrito una realidad de hecho por quien tenga fe pública o atribución para ello. Fijar, señalar con certeza [17].

Algunos autores y muchas empresas en general definen certificación como: *Acto de conferir legitimidad, aprobación, garantía formal o acto de conceder crédito o reconocimiento* [11]. Esta garantía escrita a menudo se establece en forma de una marca de certificación o etiqueta aplicada al producto o a su documentación, y/o un registro que está disponible en un organismo público.

La **certificación de hardware** es el procedimiento por parte de una tercera parte (neutral), ya sea una empresa o institución estatal que valide que un hardware está apto para la instalación de un software en un sistema determinado e interactúa de forma correcta con este [12].

1.2.2 Software Libre

Según la Fundación del Software Libre (FSF, por sus siglas en inglés), organización sin fines de lucro fundada por Richard M. Stallman en el año 1984, los programas libres se definen, según sus términos de licenciamiento[9] como aquellos que:

- Se pueden usar con cualquier propósito.
- Se pueden estudiar y adaptar a las propias necesidades.
- Se pueden copiar y redistribuir versiones idénticas.
- Se pueden mejorar y redistribuir las versiones mejoradas.

El SWL suele estar disponible gratuitamente, está compuesto por varias comunidades donde los usuarios comparten sus desarrollos por un bien común, aunque no es obligatorio que sea así. Por el contrario, el software gratuito (freeware, adware) no es libre: un usuario lo puede usar gratuitamente, pero no es estrictamente suyo, no tiene garantizado todos los derechos citados. Los programas de Código Abierto (Open Source) parten de una filosofía diferente, aunque desaparezca la ambigüedad de la palabra inglesa "free" (al mismo tiempo "libre" y "gratuito"), no da, en muchos casos, libertad de distribución de las modificaciones a los usuarios y restringe su uso comercial[24].

La migración al software libre bajo la premisa *Free and Open Source Software* (FOSS) avanza muy rápidamente, a escala mundial, en el caso de los servidores de ficheros en general, incluyendo la creciente utilización de las máquinas virtuales. El modelo de negocio del SWL se fundamenta en la venta de servicios de valor añadido sobre productos, que es el verdadero negocio rentable y no el modelo de cobro de licencias de software que las grandes corporaciones implementan con una situación casi de monopolio [3].

1.3 Tendencias actuales relacionadas al hardware

El hardware cada vez más se está integrando al uso cotidiano, en las grandes empresas su uso es indispensable, pues compañías como Facebook, Google o Microsoft no existieran de no ser posible el desarrollo actual de este. Su uso ha suscitado una polémica muy grande a nivel mundial, debido al auge y la aceptación que está teniendo el SWL en las empresas, desde Pequeñas y Medianas Empresas (PYMES) hasta compañías de gran envergadura como IBM, Samsung, Google, Dell entre otros, que lo implementan en servidores, teléfonos, aplicaciones y accesorios electrónicos de uso personal y doméstico. Cada día son más personas e instituciones que se suman al uso de hardware con SWL, aparejado a esto existen situaciones que van en contra de la opinión generalizada de que el hardware funcionaba perfectamente tanto en el software propietario como en el libre[24].

En ocasiones algunos fabricantes de hardware crean sus productos poniendo restricciones con el fin de que estos sólo puedan ser utilizados con determinado software propietario que desarrolla la empresa con la que tienen negocios. Los perjudicados son los usuarios finales, que utilizan SWL, los cuales tienen que crear sus propios controladores de hardware, que en ocasiones no funcionan de la manera más óptima pues la gran mayoría del hardware es cerrado, sin acceso a su diseño[6].

Bajo la presión de las compañías discográficas y cinematográficas, el software que la gente puede utilizar está siendo diseñado cada vez más específicamente para restringirlos. Existe una funcionalidad maliciosa denominada como DRM o «Gestión de Restricciones Digitales» [5, 6] y es la antítesis, en espíritu, a la libertad que el SWL busca proveer. Y no sólo en espíritu: puesto que el objetivo del DRM es dificultar, hacer imposible o incluso ilegal modificar los programas que lo implementan. Empresas como Microsoft, Sony, Atari y BBC son ejemplos de compañías que defienden y aplican el DRM, tanto en el hardware como a los contenidos digitales que crean y lo más preocupante es que estos mecanismos de restricción están siendo incluidos en todo tipo de dispositivos sin informar a quienes los adquieren, bajo el nombre de "Trusted Computing" o computación confiable como engañosamente se le quiere hacer ver a las personas. Se pueden encontrar en computadoras, teléfonos, televisores, radios, juguetes, fotocopiadoras, impresoras y muchos más[6, 24].

Algunas versiones de computación traicionera como es llamada por los seguidores del SWL requieren que el sistema operativo esté específicamente autorizado por una compañía en particular. Los sistemas operativos libres no podrían instalarse. Algunas versiones de computación traicionera necesitarán que cada programa sea específicamente autorizado por el desarrollador del sistema operativo.

Prácticamente todas las computadoras que se compran tienen Windows preinstalado, pero no por elección, Microsoft dicta los requisitos a los vendedores de hardware, quienes no ofrecen PCs sin Windows instalado, a pesar de que mucha gente los pide. Incluso las computadoras disponibles con sistemas operativos como GNU/Linux preinstalado a menudo han tenido Windows instalado primero. A Microsoft se le ha encontrado culpable de hacer monopolio en todo el mundo[13], con Windows Vista, Microsoft trabajó con los fabricantes de PC para aumentar significativamente las especificaciones de hardware para la experiencia del usuario promedio, haciendo que la gente tuviera que comprar computadoras nuevas que funcionaran con el sistema operativo actualizado.

La tivoización del hardware es lo que ha sido denominado por la FSF como la utilización de un sistema operativo libre en hardware restrictivo, o sea que cualquier actualización o modificación que se le realice al software, el hardware la rechazará dejándola inoperativa, por lo tanto, se arruina el propósito de la licencia GPL de mantener las libertades del software.

En el año 2006 la compañía TiVo comenzó a usar GNU/Linux y SWL bajo la licencia GPLv2.0 en sus dispositivos de grabación de video digital. Pero repentinamente hizo una acción que tomó por sorpresa al propio Richard Stallman, utilizó un hardware restrictivo que impide que los usuarios puedan modificar el código fuente e instalar sus propias

aplicaciones, de ahí surge el fenómeno de la “tivoización” del hardware, que explota una brecha legal en la licencia que aplica y que fue corregida con la GPLv3.0 presentada en el 2007 [4].

Muchas de las prácticas anteriormente expuestas traen consigo que la interoperabilidad del hardware con los sistemas operativos libres se vea amenazada, ya sea directamente por restricciones de software implementadas en el hardware como el DRM o por la ausencia de drivers. Muchos usuarios para poder usar dispositivos en estas circunstancias, deben hacer extensas búsquedas de información sobre cómo solucionar la compatibilidad de los mismos. A partir de estas situaciones, empresas que contribuyen al desarrollo del SWL crean mecanismos para acumular la información de estas experiencias y exponen sus resultados en extensas listas de compatibilidad de hardware.

1.4 Sistemas de certificación de hardware para Software Libre

El amplio desarrollo de las tecnologías ha propiciado que cada vez se disponga de mayor acceso a nuevos dispositivos electrónicos, para mejorar la calidad de vida o agilizar tareas que comúnmente eran difíciles de realizar de la manera tradicional. Por ende es muy común que cuando se conecta un nuevo dispositivo a una computadora o hardware, que no dispone de controladores actualizados (drivers), no se pueda utilizar. Esto es debido a que probablemente el dispositivo no esté certificado por el fabricante para determinado sistema operativo por ser muy nuevo o porque específicamente no se le de soporte. Para empresas como IBM, que proveen un sin número de servicios es inadmisible y costoso parar alguno de estos, por incompatibilidades del hardware con el sistema operativo.

La certificación de hardware es una necesidad para una correcta migración a una nueva plataforma tecnológica, para validar o comprobar que el hardware del que se dispone es compatible con el software. Muchas compañías se han trazado la estrategia de crear centros de certificación de hardware para certificar el hardware compatible con los distintos sistemas operativos existentes. Empresas como Red Hat, The Linux Foundation, Canonical, Microsoft o en general empresas que desarrollan hardware como HP, CISCO y AMD tienen centros de certificación con ingenieros y trabajadores calificados para hacer pruebas y certificar hardware. Como complemento de estos centros de certificación se crean los sistemas de certificación de hardware para automatizar el proceso de certificación y proveer a los clientes la información referente a la compatibilidad de sus dispositivos.

Muchas empresas liberan las denominadas HCL (Hardware Compatibility List), que por lo general son listados de

compatibilidad de hardware que permiten hacer búsquedas de componentes. Presentan diseños simples como es el caso de Debian HCL, OpenSuse HCL y sitios como Linux-drivers u Openprinting. En sentido general las opciones de los sistemas de certificación para SWL son muchas y con una gran variedad de iniciativas. Una funcionalidad muy necesaria a la hora de implementar una migración a SWL es la búsqueda de hardware homólogo, la cual no está presente dentro de los sistemas analizados en el transcurso de la presente investigación.

Los sistemas estudiados que se analizan a continuación, a pesar de estar enfocados al software libre, no permiten su uso fuera de sus instituciones debido a sus licencias de carácter propietario e interno. Esta restricción imposibilita el uso de los mismos como solución para el problema que pretende resolver la presente investigación. Sin embargo, el análisis de los mismos permite destacar algunas características importantes para el sistema, como lo son:

- La certificación de hardware se establece mediante categorías.
- La certificación es avalada por pruebas registradas en documentos formales de certificación de la institución.
- Los usuarios pueden comentar el estado de compatibilidad.
- Evaluación de compatibilidad mediante valoraciones de los usuarios.
- El hardware es ordenado por categorías.
- Uso de filtros en los listados de hardware.
- Integración con agentes de extracción de datos de hardware.
- Funcionalidad de búsqueda de hardware mediante: fabricante, modelo y versión.
- Los usuarios pueden importar desde reportes de hardware.

A continuación se analizan los principales sistemas de certificación de hardware para sistemas operativos libres con más impacto a nivel mundial.

1.4.1 Sitio web de certificación de hardware de Ubuntu

El sistema de certificación de Ubuntu (https://launchpad.net/hardware-certification) desarrollado con un CGI en python, cuenta con un diseño gráfico simple, tiene un amplio catálogo de hardware en el cual las certificaciones pueden ser accedidas mediante fabricantes, modelos o las versiones del sistema operativo. Además tiene un catálogo por componentes ordenado por categorías y presenta una buena arquitectura de la información, que

permite una navegación rápida y eficaz. Permite mostrar un listado del hardware recientemente certificado y hacer búsquedas dentro del catálogo.

El Centro de Certificación de Ubuntu brinda varios servicios de certificación a las compañías de hardware, presenta dos programas de certificación:

Ubuntu Certified es para las empresas de hardware asociadas a Canonical, donde estas deben enviar los productos para ser testeados y certificados por los ingenieros de Canonical. Las actualizaciones son incluidas en las versiones estables de Ubuntu y tienen mayor prioridad de revisión.

Works with Ubuntu es una designación para hardware periférico externo al sistema, como impresoras, escáner y dispositivos USB.

El sistema es parte de una plataforma de certificación de hardware que presenta un sistema de testeo del proyecto Checkbox (https://launchpad.net/checkbox), una aplicación cliente de testeo de hardware que se incluye en la distribución[1] de Ubuntu que puede ser lanzada por el usuario con la aplicación checkbox-gtk [2]. En general es un sistema con un amplio programa de certificación y una buena cantidad de ingenieros encargados de realizar las comprobaciones y las certificaciones.

1.4.2 Centro de certificación de hardware de Red Hat

El centro de Red Hat es una iniciativa totalmente comercial, tiene como objetivo fundamental establecer relaciones con los principales vendedores de hardware, proporcionándole la posibilidad de incluir sus productos dentro del catálogo de hardware certificado. Inicialmente presenta la navegación del catálogo delimitada por las versiones de la distribución de Red Hat, a partir de las cuales se puede navegar por las certificaciones mediante sistemas o por componentes. Permite hacer búsquedas simples o avanzadas, las certificaciones admiten notas que son artículos de la Base de Conocimiento de Red Hat que se aplican a determinado hardware con impacto en el nivel de soporte ofrecido.

Carece de funcionalidades que permitan a los usuarios intercambiar experiencias relacionadas al funcionamiento del hardware y este sistema operativo. La información es muy confiable por el alto grado de profesionalismo y legalidad en

[1] Una distribución Linux o distribución GNU/Linux (coloquialmente llamadas distros.), es una distribución de software basada en el núcleo Linux que incluye determinados paquetes de software.

que se ampara el sistema.

En el proceso de certificación los principales requisitos son:

- Se certifican los modelos de hardware, no las especificaciones de configuración del modelo. Toda la configuración de hardware opcional designada bajo el mismo modelo debe ser probada nuevamente.
- Las pruebas se realizan con un estándar de instalación de Red Hat Enterprise Linux diseñado especialmente para ello.
- La certificación se otorga en contra de una versión específica y la arquitectura.

1.4.3 Sistema automatizado para la selección de hardware compatible con SWL

Es un sistema implementado en la UCI desarrollado con el CMS Drupal, presenta una base de datos poco flexible a la hora de buscar homólogos por características de los componentes, maneja poca información del hardware, gestiona las certificaciones mediante reportes de compatibilidad y permite realizar comentarios sobre los reportes y el hardware [4].

1.4.4 Hardware4Linux

Hardware4Linux es un sistema implementado en el framework web Django, es una interesante iniciativa en la cual participan varios colaboradores en todo el mundo que no está centrada en una única distribución, presenta un diseño sencillo que proporciona información relacionada con la compatibilidad de hardware con SWL. Permite a los usuarios de cualquier distribución reportar sus experiencias con el hardware en general mediante comentarios y un sistema de votaciones. Ofrece un software de forma libre que se instala en la máquina y que ayuda a realizar reportes del hardware donde se está ejecutando. Muestra una variada información del hardware, permite acceder a la información mediante lectores RSS y muestra estadísticas por fabricantes generando gráficas mediante varios criterios de compatibilidad. Carece de un sistema de búsqueda por criterios.

1.5 Metodología de desarrollo

La Metodología definida para el sistema es SXP debido a que está encaminada a una programación rápida con iteraciones incrementales y ciclos cortos y es la metodología que se aplica en el proyecto SIMAYS del cual es parte esta solución. A continuación se hace una breve caracterización de la misma.

SXP

Metodología ágil desarrollada en la UCI, que hace uso de SCRUM para la gestión del trabajo y toma de XP las mejores prácticas que guían el desarrollo del software, como la refactorización y pruebas continuas. Ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software, para el mejoramiento de la actividad productiva. Fomenta el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo. Ayuda al líder del proyecto a tener un mejor control y cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto.

SXP consta de cuatro fases:

Planificación-Definición

Se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto. Cuenta con tres tareas principales: ingeniería de sistemas o de información, planificación del proyecto y análisis de los requisitos.

Desarrollo

Se implementa un sistema listo para entregar en una serie de iteraciones. Cuenta con tres tareas fundamentales: diseño del software, generación de código y prueba del software.

Entrega

Se despliega y se pone en marcha el producto software.

Mantenimiento

Se realiza el soporte para el cliente.

SXP es ideal para proyectos de corta duración con requisitos cambiantes o no bien definidos, donde prevalezca la retroalimentación entre el cliente y el equipo de trabajo. El desarrollo con SXP se realiza en iteraciones cortas (sprints)

a lo largo de tres fases, dándole cumplimiento a un grupo de actividades, de las que se generan una serie de artefactos, que documentan el proceso de desarrollo, obteniendo un release del producto con nuevas funcionalidades [22].

1.6 Lenguajes, tecnologías y herramientas

1.6.1 Lenguajes de programación

PHP

PHP (PHP Hypertext Preprocessor) es un lenguaje de propósito general muy fácil de aprender con respecto a otros lenguajes utilizados para el mismo propósito, como JAVA o ASP, creado en 1994 por Rasmus Lerdorf, miembro del equipo de desarrollo de Apache. La implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la FSF considera esta licencia como software libre. Es desarrollado y mantenido por alrededor de 1000 ingenieros de PHP Development Team[10].

Características:

- Alto rendimiento.
- Gran comunidad de apoyo.
- Multiplataforma.
- Multitud de extensiones.
- Orientado a la Web.

Existe una gran variedad de aplicaciones web dinámicas desarrolladas en este lenguaje como lo son Facebook, PrestaShop, WordPress Drupal, Moodle y otras, la mayoría de las páginas web de Internet están creadas con este debido a su simplicidad, adaptabilidad, interoperabilidad, portabilidad y rendimiento.

JavaScript

JavaScript es un lenguaje de scripting basado en objetos creado por Brendan Eich, oficialmente liberado en 1995, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. JavaScript es un dialecto de ECMAScript y se caracteriza por ser un lenguaje basado en prototipos, con entrada dinámica y con funciones de primera clase. Es un lenguaje muy sencillo de aprender, aumenta la usabilidad de las páginas web, reduce la carga del servidor empleándolo en validaciones o usando tecnologías como AJAX (Asynchronous JavaScript And XML).

1.6.2 Lenguajes de marcado

HTML

HTML (HyperText Markup Language), lenguaje de marcado de hipertexto, estándar para describir el contenido y la apariencia de las páginas creado en 1991 por Berners-Lee, estandarizado por la W3C[2]. Es el lenguaje utilizado para crear y reconocer documentos hipermedia a los que se accede a través de la red Internet. Este lenguaje consiste en un sistema de introducción de marcas de referencia en un fichero de texto, basado en el estilo SGML, utilizado para la creación de documentos de hipertexto, que en Internet generalmente poseen la extensión "html".

XML

XML (eXtensible Markup Language) es un subconjunto simplificado del SGML[3] el cual fue diseñado principalmente para documentos Web. Es un metalenguaje extensible de etiquetas desarrollado por el W3C que permite definir la gramática de lenguajes específicos.

[2]Organización internacional que desarrolla y da soporte a los estándares de las principales tecnologías Web.

[3]Lenguaje de marcado generalizado, para la organización y etiquetado de documentos.

YAML

YAML (YAML Ain't Markup Language) es un lenguaje de serialización de datos propuesto por Clark Evans en el 2001,con una sintaxis muy simple que se usa para describir los datos. Su uso se extiende a múltiples aplicaciones, principalmente en ficheros de configuración e intercambio ligero de información y como lenguaje para la representación del esquema de base de datos para el ORM (Object Relational Mapping) Doctrine.

1.6.3 Lenguaje de modelado

UML

Lenguaje Unificado de Modelado (UML): Lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema software. Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo.

1.6.4 Tecnologías

Framework para el lenguaje PHP

El amplio y extendido desarrollo de Internet ha proporcionado aplicaciones web cada vez más complejas, lo que ha provocado que al desarrollarlas se complejice la arquitectura y se aplique mayor cantidad de tecnologías. Los frameworks de desarrollo agilizan el proceso creando un marco de trabajo sencillo utilizando las mejores prácticas de programación. Para el desarrollo de la solución se emplea Symfony debido a ser uno de los requisitos del sistema, además de ser una de las mejores opciones en cuanto a frameworks para PHP. A continuación se hace una breve referencia del mismo.

Symfony

Symfony es un completo framework creado por Sensio Labs, una aplicación Open Source bajo la Licencia MIT, diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica

de servidor y la presentación de la aplicación web mediante el Patrón Modelo-Vista-Controlador (MVC). Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja y automatiza las tareas más comunes. Desarrollado completamente con PHP 5, ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel.

Características:

- Adaptable a políticas y arquitecturas diferentes.
- Alta seguridad.
- Extensible mediante librerías o plugins.
- Gestión del caché de la aplicación.
- Independiente del SGBD mediante ORM personalizado.
- Mantenimiento sencillo y código estandarizado.
- Sigue la mayoría de las mejores prácticas y patrones de diseño web.
- Testeo unitario y funcional automatizado.

Symfony se usa en Yahoo! Answers, Daily Motion, Delicious, por su alta escalabilidad, Soporte de Larga Duración (LTS, por sus siglas en inglés) durante tres años, seguro y una alta calidad con más de 9000 pruebas unitarias.

Plugins y librerías utilizadas

jQuery

Es una biblioteca o framework de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código.

- Compatibilidad con la mayoría de los navegadores.
- Efectos y animaciones dinámicas.
- Gestión de Eventos.
- Manipulación de la hoja de estilos CSS.

- Soporte de extensiones.

Es un producto serio, estable, documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework. Tiene una dilatada comunidad de creadores de plugins o componentes, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar interfaces de usuario, galerías, votaciones, efectos diversos, etc.

Librería jQuery UI

Es una librería de componentes para el framework jQuery que provee un conjunto de plugins, efectos visuales y widgets. Su utilización permite construir aplicaciones web altamente interactivas y con soporte para varios navegadores web. Incluye clases que cubren un amplio rango de necesidades comunes de las interfaces de usuario y puede ser manipulado usando jQuery Theme Roller. Permite aplicar estilos correctores, estructurales y basados en temas (colores, fuentes, fondos, etc.).

Plugin ckWebServicePlugin

El plugin ckWebServicePlugin permite construir una API de servicios web para las aplicaciones en Symfony. Viene con un potente y fácil generador de WSDL compatibles con WS-I, para un máximo de interoperabilidad con clientes PHP, .NET y Java. Utiliza la extensión nativa SOAP de PHP.

Librería Zend Lucene

Es una librería del Zend Framework de código abierto escrita completamente con PHP 5. Zend Lucene es un buscador genérico de texto, que guarda sus índices en archivos y no requiere de un servidor de base de datos, útil para cualquier aplicación que requiera indexado y búsqueda a texto completo.

Características:

- Búsqueda por ranking, que muestra primero los mejores resultados.
- Búsqueda por un campo específico, como por ejemplo título, autor o contenidos.
- Soporta consultas: mediante frases, booleanas, con comodines, de proximidad, basadas en rangos y muchos otros tipos de consultas.

El centro de la arquitectura lógica de Lucene se encuentra en el concepto de Documento (Document) que contiene

Campos (Fields) de texto. Esta flexibilidad permite a Lucene ser independiente del formato del fichero.

Servicios web

EL principal uso de Internet hoy es el acceso interactivo a documentos y aplicaciones. La Web crece significativamente y conjuntamente a esto es necesaria mayor interoperabilidad entre las distintas tecnologías para lograr mayor comunicación entre diferentes plataformas. Aquí es donde entra a desempeñar un papel fundamental los servicios web, los cuales permiten elevar la interoperabilidad a su mayor expresión permitiendo el acceso a los datos, funcionalidades y sistemas haciendo transparentes en la comunicación la forma que están implementadas las fuentes. Un servicio web es un servicio accesible a través de una red, generalmente Internet que puede ser descrito, publicado, localizado, e invocado, que usa un sistema de mensaje estandarizado y no está atado a un sistema operativo o lenguaje de programación [28].

La arquitectura básica del modelo describe un consumidor, un proveedor y ocasionalmente un corredor (broker) y relacionados con estos agentes está las operaciones de publish (publicar), find (encontrar) y bind (enlazar).

Un servicio web está definido por la siguiente pila de protocolos (véase, figura A.1 en anexos):

Service transport Capa responsable del transporte de mensajes entre aplicaciones a través de los protocolos HTTP, SMTP, FTP, y BEEP.

XML messaging Capa responsable de la codificación de mensajes en un formato común XML para que los mensajes se puedan entender por las aplicaciones. que se comunican. Actualmente, esta capa incluye XML-RPC, SOAP y REST.

Service description Capa responsable de describir la interfaz pública del servicio web específico, es manejada por WSDL.

Service discovery Capa responsable de centralizar los servicios en un registro común, y proporcionando facilidad de publicación y búsqueda de los mismos mediante UDDI.

UDDI (Universal Description, Discovery, and Integration) es un modelo de directorios para servicios web, cuyo objetivo es ser accedido por los clientes y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios web del catálogo de registros.

WSDL (Web Service Description Language), una gramática XML utilizada para describir servicios web. Define como

el servicio web es accedido, las operaciones que ejecuta, como son pasados los mensajes y su estructura. No es requerido para trabajar con SOAP, es una parte integral de las especificaciones WS-I Basic Profile,[4] de la organización WS-I (Web Services Interoperability) [19].

En el desarrollo de la presente investigación se utiliza el protocolo de comunicación SOAP por ser robusto y fácil de consumir. Además de ser el protocolo definido para la comunicación de la PMSWL. Seguidamente se describe en más detalle.

SOAP

Simple Object Access Protocol (SOAP) es un protocolo que permite la comunicación entre aplicaciones a través de mensajes por medio de Internet, independiente de la plataforma, y del lenguaje. Está basado en XML y es la base principal de los servicios web.

SOAP ha tenido una gran aceptación en el mundo dado el impulso que le han dado compañías como Microsoft, IBM, SAP y Ariba. Esta inusual colaboración entre empresas multinacionales ha provocado un acelerado uso en entornos empresariales con grandes necesidades de interoperabilidad.

Características:

Interoperabilidad Su desarrollo sobre los estándares permite que las aplicaciones puedan comunicarse en múltiples entornos.

Independiente del lenguaje SOAP no especifica una API de programación, por lo que su implementación se puede crear en cualquier lenguaje programación, como en Java o una plataforma como Microsoft .Net.

Independiente del protocolo de transporte La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.

Utiliza estándares internacionales La especificación SOAP se extiende de los estándares existentes, constantemente probados en todo el mundo.

[4] Conjunto de especificaciones de servicios web no propietarios, refinamientos, aclaraciones y aplicaciones de especificaciones que promueven interoperabilidad.

1.6.5 Herramientas para el desarrollo

Para el entorno de desarrollo se listan a continuación las herramientas que se seleccionaron, para ello se hizo una caracterización de sus principales funcionalidades. Teniendo en cuenta el soporte de la herramienta y licenciamiento (preferiblemente SWL u Open Source). En el caso de Visual Paradigm aunque su licenciamiento no es SWL, se opta por esta herramienta porque es la herramienta CASE[5] más completa que se dispone para GNU/Linux.

Netbeans

Netbeans IDE (Integrated Development Environment) es un entorno de desarrollo distribuido bajo la doble Licencia CDDL (Commom Development and Distribution License) y la GPL versión 2 con Classpath Exception. Posee soporte para PHP, JavaScript y características visuales para el desarrollo web. Es un producto libre, multiplataforma y gratuito sin restricciones de uso.

Características:

- Integración nativa con sistemas gestores de bases de datos (MySQL y PostgreSQL).
- Orientación a servicios web y modelado UML.
- Soporte a frameworks como Hibernate, Struts, Symfony, Spring entre otros.
- Soporte de CVS (Control Version System).
- Soporte para varios lenguajes de programación como PHP, JavaScript, Python, Java entre otros.

PostgreSQL

Es un avanzado sistema de gestión de bases de datos relacionales, accesible a un amplio rango de plataformas. Uno de sus mayores beneficios es el ser un software Open Source, publicado bajo la licencia TLP (The PostgreeSQL License) muy similar a la licencia BSD (Berkeley Distribution Software). Es un sistema que requiere pequeños o casi ningún mantenimiento, y provee un costo bajo por propiedad, además de que cuenta de una versión comunitaria libre de pago.

Tiene un amplio rango de funcionalidades, con más de 20 años de desarrollo continuo. Fue originalmente desarrollada

[5]Ingeniería de Software Asistida por Ordenador.

por el grupo de investigación de bases de datos de la universidad de Berkeley, actualmente es mantenida por un gran grupo de desarrolladores y contribuyentes, de los cuales muchos tienen trabajos a tiempo completo como diseñadores, desarrolladores y administradores de bases de datos.

Principales características:

- Arquitectura Cliente-Servidor.
- Buen diseño concurrente en su arquitectura interna de lectores/escritores.
- Compilación excelente de Estándares SQL hasta SQL 2008.
- Implementa el uso de rollback[6], subconsultas y transacciones.
- Muy extensible y configurable para varios tipos de aplicaciones.
- Soporte para integridad referencial.

Una de las características estrellas de Oracle a partir de Oracle 7 es la conocida "snapshot isolation", donde lectores no bloquean escritores y viceversa. PostgreSQL fue la primera base de datos Open Source en implementar esta funcionalidad y ofrecer una completa implementación [20] mediante un sistema denominado Acceso Concurrente Multiversión (MVCC, por sus siglas en inglés), el que permite 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 otros gestores, eliminando la necesidad del uso de bloqueos explícitos.

Es usado por un sin número de compañías, dentro de las que se encuentran Apple, BASF, CISCO, IMDB.com, Skype, Yahoo, entre otras.

Apache Web Server

Apache es el servidor web por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Surge a partir de un servidor web de la NCSA (National Center for Super Computing Applications) como corrección a errores y mejoras importantes al producto inicial. En 1999 los desarrolladores fundan la Apache Software Foundation, organización que da soporte legal al desarrollo de Apache y que

[6] Operación que devuelve a la base de datos a algún estado previo.

lo mantiene actualmente bajo la licencia Apache Licence 2.0, que lo convierte en software libre.

Principales características:

- Admite servidores virtuales mediante IP y nombres virtuales.
- Alta configurabilidad en la creación y gestión de logs.
- Módulos multiproceso.
- Multiplataforma.
- Reescritura y corrección de URL.
- Simplicidad de configuración.
- Soporte para FastCGI y CGI.
- Soporte de Server Side Include (SSI) y Secured Socket Layer (SSL).

Visual Paradigm para UML

Visual Paradigm for UML (VP-UML) es una herramienta de diseño UML y herramienta CASE UML diseñada para la ayuda al desarrollo de software. VP-UML soporta los principales estándares de la industria tales como Lenguaje de Modelado Unificado (UML), SysML, BPMN, XMI, etc. Ofrece un completo conjunto de herramientas a los equipos de desarrollo de software necesario para los requisitos de la captura, software de planificación, la planificación de controles, el modelado de clases, modelado de datos, etc. Proporciona un ambiente de modelado visual para satisfacer la tecnología del software y necesidades de comunicación. Es muy fácil de usar y presenta un ambiente gráfico agradable para el usuario.

Subversion

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS. Es software libre bajo una licencia de tipo Apache/ BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo. La capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración.

Características:

- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente; Tiene costo de complejidad constante $O_{(1)}$ y no lineal $O_{(n)}$ como en CVS.
- Maneja eficientemente archivos binarios.
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.
- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Se envían sólo las diferencias en ambas direcciones.

1.7 Conclusiones

En este capítulo se realiza un estudio de los principales sistemas de certificación internacionalmente y en el país, los cuales sirven como referencia en cuanto a funcionalidades y diseño, pues no se adaptan a los requerimientos del sistema que se desarrolla. Se abordan conceptos fundamentales como la certificación y homologación de hardware, actualidad del SWL y las tendencias alrededor del hardware, necesarios para comprender la actualidad y necesidad de la solución propuesta. Se caracterizan elementos de la metodología de desarrollo SXP y las principales herramientas que complementan el proceso de desarrollo.

Las tecnologías expuestas forman parte fundamental del desarrollo permitiendo mayor flexibilidad e interoperabilidad, siendo los servicios web el paradigma más importante como modelo para la integración con la PMSWL. En el sistema se utiliza el protocolo SOAP para la capa de mensajería, el lenguaje XML para la codificación y WSDL para su descripción, por ser requisitos de la PMSWL.

2

Definición del sistema

2.1 Introducción

En el presente capítulo es descrito el funcionamiento del sistema, estableciendo los requisitos técnicos y operacionales que debe contemplar la solución, recogiendo las funcionalidades del sistema de certificación de hardware reflejadas en las historias de usuarios. Se describen y modelan los procesos mediante IDEF0 para una mayor comprensión de la manera en que se desarrollan. En la metodología SXP se propone diseñar la solución más simple que pueda funcionar, al definir no se enfatiza en una arquitectura estable para el sistema, sino que esta se asume de forma evolutiva. Por tanto para solventar los inconvenientes que pudiera generar el no contar con dicha arquitectura, se utiliza el diseño con metáforas [1], que da como resultado el modelo de diseño. [21]. Se define el diseño de la arquitectura y la propuesta de solución para el problema planteado en la introducción.

2.2 Solución propuesta

Cuando se inicia el proceso de migración en una institución, dentro de la PMSWL se crea un nuevo proyecto de migración en el sistema de planificación, control y seguimiento, lo que comienza la etapa de preparación definida en la metodología de migración. Comienza entonces el flujo de trabajo Evaluación en el cual se realizan las tareas de recopilación de datos, inicialmente interviene el sistema LimeSurvey, el cual se encarga de automatizar las encuestas en la institución, permitiendo la identificación de los tipos de usuarios que participan en el proceso de migración.

Posteriormente se procede a la extracción de la información del software y el hardware mediante la herramienta GITI

[1] Una metáfora es una historia compartida que describe cómo debería funcionar el sistema.

de cada una de las computadoras existentes en la entidad. El Módulo de Automatización de la Gestión de la Migración (MAGM) a partir de los datos importados por GITI efectúa la asociación (los datos: software y dispositivos de hardware) de los sistemas respectivos: Directorio de Software en línea y SICEH. Mediante MAGM y SICEH es posible poder determinar la cantidad de hardware compatible con Nova, para así proseguir con los demás procesos de la migración.

Para poder llevar a cabo lo anteriormente expuesto es necesario contar con una extensa base de datos de hardware certificado dentro de SICEH, información que es actualizada mediante el agente de recolección de información de hardware SistDCH. El hardware es certificado por el especialista en migración, encargado de ejecutar las tareas de las etapas de migración definidas en la metodología de migración.

Como propuesta de la solución, se crea SICEH como un componente de la PMSWL y un módulo para el mismo, de interfaz de usuario dentro del Sistema de Interfaces Único de la PMSWL (SINPM). La comunicación entre SICEH y el módulo se establece mediante el protocolo SOAP mediante la publicación de los servicios de SICEH. Asegurando en todo momento la seguridad mediante el Sistema de Control de Seguridad de la PMSWL (SCS), al cual por cada petición del módulo de interfaz se le hace una comprobación de autorización del servicio solicitado.

Dentro de SICEH se definieron los siguientes módulos para brindar los servicios:

arch Gestiona las arquitecturas de los sistemas operativos.

category Gestiona las categorías del hardware del sistema.

certificationCategory Gestiona las categorías de las certificaciones de hardware.

comment Gestiona los comentarios de compatibilidad de la relación entre hardware y sistema operativo.

compatibility Gestiona las relaciones de compatibilidad entre hardware, sistema operativo y drivers.

distribution Gestiona las distribuciones de los sistemas operativos.

driver Gestiona los drivers en el sistema.

hardware Gestiona el hardware en el sistema.

manufacturer Gestiona los fabricantes de hardware.

notification Gestiona las notificaciones, generadas por eventos priorizados como lo son: la importación de hardware o eliminación de una certificación.

operativeSystem Gestiona los sistemas operativos en el sistema.

rating Gestiona las valoraciones de la compatibilidad.

whitelist Gestiona las reglas de asociación de las categorías externas de importación.

2.3 Modelo del negocio

En la migración a SWL, cuando se analiza la compatibilidad del hardware intervienen dos procesos fundamentales que son llevados a cabo por el especialista en migración. La certificación y la homologación de hardware son procesos separados que pueden ser utilizados durante la migración de una empresa o pueden ser solicitados directamente por una entidad externa. La certificación de hardware puede desencadenar el proceso de homologación en el caso de que no existan controladores válidos para el hardware que se está certificando, siempre y cuando la empresa cuente con los recursos necesarios para adquirir un hardware homólogo al que pretendía certificar. A continuación representan los procesos anteriormente expuestos modelados mediante IDEF0 y posteriormente se describen los mismos. Ambos procesos pueden ser vistos en más detalle en la figura A.3 y A.4, en los anexos.

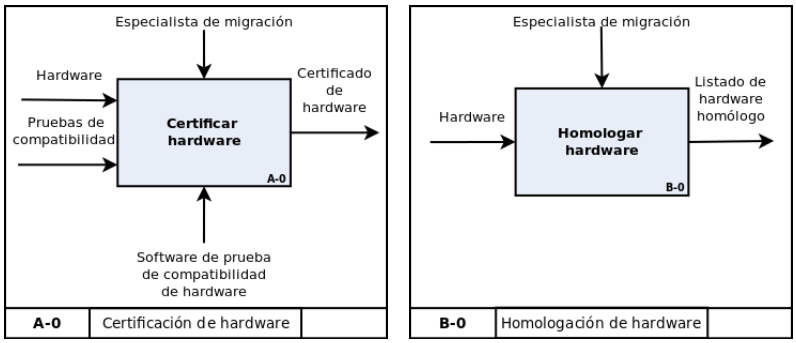

Figura 2.1: Procesos de certificación y homologación de hardware

A-0 Certificación de hardware: tiene como entrada el hardware que va a ser analizado para certificar, controlado por el especialista en migración dando como salida el certificado de hardware. Para probar la compatibilidad el encargado de ejecutar la certificación puede auxiliarse de un software de prueba de compatibilidad de hardware como mecanismo, para un análisis más detallado.

B-0 Homologación de hardware: tiene como entrada el hardware que va a ser homologado, controlado en todo momento por el especialista en migración. Como salida se devuelve un listado de hardware certificado y homólogo al hardware de la entrada.

2.4 Captura de requisitos

> "La ingeniería de requisitos es el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener con un costo reducido el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo de un sistema que satisfaga las necesidades del usuario. Tenga en cuenta que todas las disciplinas de la ingeniería son semejantes, la ingeniería de requisitos no se guía por conductas esporádicas, aleatorias o por modas pasajeras, sino que se debe basar en el uso sistemático de aproximaciones contrastadas" [16].

2.4.1 Lista de Reserva del Producto

Es una lista priorizada que define el trabajo que se va a realizar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, suelen surgir los más importantes que casi siempre son más que suficientes para un sprint (iteración) [21].

Requisitos funcionales

Los requisitos funcionales son condiciones o capacidades que el sistema debe cumplir. Se emplean para especificar los principales servicios que el sistema proporciona. Determinan el comportamiento de la aplicación ante entradas y salidas específicas sin alterar las funcionalidades. En la siguiente tabla se recogen en cada una de las filas los requisitos identificados del producto. Se muestra el número identificativo del requisito, la descripción del requisito, el tiempo estimado para su cumplimiento y el nombre del estimador respectivamente.

RF (Requisitos funcionales)				
Prioridad	**Ítem***	**Descripción**	**Estimación**	**Estimado por**
Sistema: Certificación y Homologación de Hardware				
Alta				
	1	Adicionar fabricante	4 horas	Ricardo Carbajo Pérez

Prioridad	Ítem*	Descripción	Estimación	Estimado por
	2	Adicionar categoría de hardware	3 horas	
	3	Adicionar hardware	4 horas	
	4	Editar información de hardware	4 horas	
	5	Eliminar hardware	1 hora	
	6	Mostrar información de hardware	4 horas	
	7	Mostrar listado de hardware mediante filtros	4 horas	
	8	Adicionar sistema operativo	4 horas	
	9	Adicionar driver	4 horas	
	10	Adicionar distribución de sistema operativo	2 horas	
	11	Adicionar arquitectura de sistema operativo	2 horas	
	12	Adicionar categoría de certificado	6 horas	
	13	Adicionar certificado de hardware	5 horas	
	14	Editar información de certificación	3 horas	
	15	Eliminar certificación	1 hora	
	16	Mostrar listado de certificados de hardware mediante filtros	4 horas	
	17	Mostrar información de certificado de hardware	2 horas	
	18	Adicionar regla de importación	2 horas	
	19	Buscar hardware	8 horas	
	20	Importar reporte de hardware	6 horas	
	21	Buscar hardware homólogo	4 horas	
Media				
	22	Editar información de fabricante	3 horas	Ricardo Carbajo Pérez
	23	Eliminar fabricante	1 hora	
	24	Mostrar listado de fabricantes mediante filtros	2 horas	
	25	Editar información de categoría de hardware	2 horas	

Prioridad	Ítem*	Descripción	Estimación	Estimado por
	26	Eliminar categoría de hardware	1 hora	
	27	Mostrar listado de categorías de hardware mediante filtros	2 horas	
	28	Editar información de distribución de sistema operativo	3 horas	
	29	Eliminar distribución de sistema operativo	1 hora	
	30	Mostrar listado de distribuciones de sistemas operativos mediante filtros	2 horas	
	31	Editar información de la arquitectura de un sistema operativo	3 horas	
	32	Eliminar arquitectura de sistema operativo	1 hora	
	33	Mostrar listado de las arquitecturas de los sistemas operativos mediante filtros	1 hora	
	34	Editar información de sistema operativo	3 horas	
	35	Eliminar sistema operativo	1 hora	
	36	Mostrar listado de sistemas operativos mediante filtros	8 horas	
	37	Editar información de driver	3 horas	
	38	Eliminar driver	1 hora	
	39	Mostrar listado de drivers mediante filtros	5 horas	
	40	Editar información de categoría de certificado	2 horas	
	41	Eliminar categoría de certificado	1 hora	
	42	Mostrar listado de categorías de certificados mediante filtros	2 horas	
	43	Editar información de regla de importación	3 horas	
	44	Eliminar regla de importación	2 horas	

Prioridad	Ítem*	Descripción	Estimación	Estimado por
	45	Mostrar listado de reglas de importación mediante filtros	4 horas	
	46	Importar reporte de hardware en xml desde SistDCH	4 horas	
Baja				
	47	Eliminar notificación	3 horas	Ricardo Carbajo Pérez
	48	Mostrar listado de notificaciones mediante filtros	3 horas	
	49	Mostrar información de notificación	2 horas	
	50	Adicionar comentario de hardware	2 horas	
	51	Editar información de comentario de hardware	5 horas	
	52	Eliminar comentario de hardware	1 hora	
	53	Mostrar listado de comentarios de hardware	3 horas	
	54	Evaluar compatibilidad de hardware	8 horas	
	55	Descargar agente SistDCH	4 horas	

Tabla 2.1: Listado de requisitos funcionales

Requisitos no funcionales

Los requisitos no funcionales responden a propiedades o cualidades que el software debe tener para que se pueda comportar de manera atractiva, confiable y segura. Suelen estar vinculados con los requisitos funcionales y son esenciales para que el producto alcance el éxito deseado.

RNF (Requisitos no funcionales)		
Tipo	**Ítem***	**Descripción**
Sistema: Certificación y Homologación de Hardware		

Tipo	Ítem*	Descripción
Interfaz	56	La interfaz del directorio debe estar acorde con la identidad del proyecto SIMAYS
Usabilidad	57	Debe ser de fácil y rápido manejo para todos los usuarios
Rendimiento	58	El sistema debe estar concebido para el consumo mínimo de recursos
Soporte	59	Disponer para el SO GNU/Linux de los siguientes paquetes: • php5 • php5-xsl • php-apc • php-soap • php5-cli
	60	Para el SO Windows debe ser instalada la aplicación WAMP
Portabilidad	61	El sistema debe ser multiplataforma
Legal	62	Deben utilizarse tecnologías libres para el desarrollo del sistema
Software	63	Utilizar como servidor web Apache y servidor de bases de datos PostgreSQL
Seguridad	64	Asegurar la aplicación mediante el sistema de seguridad de la PMSWL
Implementación	65	El intercambio de información debe ser mediante servicios web
	66	Usar como lenguaje de programación PHP
	67	Usar el framework javascript Jquery
	68	Usar el framework php Symfony
	69	Utilizar como mecanismo de intercambio de datos entre aplicaciones el protocolo SOAP
	70	Utilizar el framework de pruebas Lime de Symfony
	71	Utilizar el plugin de servicios web para Symfony ckWebServicePlugin

Tabla 2.2: Listado de requisitos no funcionales

2.4.2 Historias de usuario

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software, lo que equivale a los casos de uso en el proceso unificado. Las mismas son escritas por los clientes como las tareas que el sistema debe hacer y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas. Son escritas en lenguaje natural, no excediendo su tamaño de unas pocas líneas de texto [21].

Gestionar fabricante

Historia de Usuario	
Número: 1	**Nombre de Historia de Usuario:** Gestionar fabricante
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 2
Prioridad en el negocio: Media	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Media	**Puntos reales:** 1 Semana
Descripción: El fabricante incluye el nombre y la dirección web del mismo. Se gestionan los fabricantes que están almacenados en el sistema, permitiendo *insertar*, *editar*, *eliminar* y *mostrar* mediante filtros cada uno de ellos.	
Observaciones: En la interfaz web de SINPM: 1. Para mostrar mediante filtros se selecciona en el menú el enlace *Fabricantes* y se muestra un listado permitiendo filtrar mediante: nombre y dirección url. 2. Para insertar un fabricante son necesarios los siguientes datos: nombre y dirección url. • Si existe el fabricante se muestra un mensaje de error. 3. Para editar la información de un fabricante, se selecciona previamente, se modifican los datos y posteriormente se actualiza el elemento. 4. Para eliminar un fabricante, se selecciona previamente y se confirma la eliminación. • Si el fabricante está relacionado con un hardware, no se elimina y se muestra un error.	
Prototipo de interfaz: Figura A.5 (ver anexos)	

Tabla 2.3: HU: Gestionar fabricante

Gestionar categorías de hardware

Historia de Usuario	
Número: 2	**Nombre de Historia de Usuario:** Gestionar categorías de hardware
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Alta	**Puntos reales:** 1 Semana

Descripción:

Las categorías de hardware permiten determinar el grupo conceptual al que pertenece determinado hardware, permitiendo agrupar sus características específicas en determinado ámbito. Cada categoría tiene una serie rasgos de un hardware como plantillas en las cuales todos los elementos de un grupo van a compartir, permitiendo comparar unos con otros, estas plantillas van a estar compuestas por varios campos en los cuales se especifica una cualidad del hardware.

Cada categoría tiene un nombre, una descripción y si es una subcategoría de otra categoría ya definida. Se gestionan las categorías permitiendo *insertar*, *mostrar* mediante filtros, *editar* y *eliminar* las mismas.

Observaciones:

En la interfaz web de SINPM:

1. Para mostrar mediante filtros se selecciona en el menú el enlace *Categorías*, submenú de *Hardware*, y se muestra un listado permitiendo filtrar mediante: nombre, si es subcategoría y la categoría padre.
2. Para insertar una categoría son necesarios los siguientes datos: nombre, categoría de la que desciende y una descripción. Además se pueden agregar, campos de plantilla de hardware para definir características específicas de determinado hardware.

- Si existe la categoría en la base de datos no se inserta y se informa que existe ya el elemento.

3. Para editar la información de una categoría, se selecciona el elemento, se modifican sus campos y se actualiza posteriormente. Además se pueden agregar, modificar y eliminar campos de plantilla de hardware para definir características específicas de determinado hardware. • Si tiene relación con uno o varios hardwares y se modifica la categoría de la que precede, se muestra un error y no se actualiza. 4. Para eliminar una categoría, se selecciona y se confirma la eliminación. • Si existe previamente, se muestra un mensaje de error. • Si la categoría tiene campos de plantilla de hardware, estos también serán eliminados. • Si tiene hardware relacionado no se elimina y se muestra un error.
Prototipo de interfaz: Figura A.6 (ver anexos)

Tabla 2.4: HU: Gestionar categorías de hardware

Gestionar hardware

Historia de Usuario	
Número: 3	**Nombre de Historia de Usuario:** Gestionar hardware
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 2.5 Semanas
Riesgo en el desarrollo: Alto	**Puntos reales:** 2 Semanas
Descripción: El hardware contiene características de interés como lo son: el nombre del fabricante, modelo, versión, la categoría y dentro de cada categoría existen elementos específicos que describen al hardware que también son editables (tabla 2.4). Cuando se edita es posible gestionar la compatibilidad que tiene con los diferentes sistemas operativos. El hardware puede ser público, lo cual permite que todos los usuarios puedan tener acceso a este, inicialmente cuando se importa debe ser actualizado como público por un usuario autorizado. Se gestiona el hardware que está almacenado en el sistema, permitiendo *insertar*, *editar*, *eliminar* y *mostrar* mediante filtros cada uno de ellos.	
Observaciones: En la interfaz web de SINPM:	

1. Para mostrar mediante filtros se selecciona en el menú el enlace *Hardware* y se muestra un listado permitiendo filtrar mediante: fabricante, modelo, versión, fecha de ultima modificación, si es de acceso público y categoría. 2. Para insertar hardware son necesarios los siguientes datos del hardware: modelo, versión, fabricante y la categoría. Según la categoría seleccionada se muestran las especificaciones técnicas del hardware para completar. • Si existe el hardware en la base de datos se informa que existe ya el elemento. 3. Para editar la información de un hardware, se selecciona previamente, se modifican los datos y posteriormente se actualiza el elemento. Se brinda la posibilidad de gestionar los sistemas operativos con los que es compatible 4. Para mostrar hardware, se selecciona el elemento y se muestra su información. 5. Para eliminar hardware, se selecciona previamente y se confirma la eliminación. • Si el hardware tiene certificados de compatibilidad, no se elimina y se muestra un error.
Prototipo de interfaz: Figura A.7 (ver anexos)

Tabla 2.5: HU: Gestionar hardware

Gestionar distribución de sistema operativo

Historia de Usuario	
Número: 4	**Nombre de Historia de Usuario:** Gestionar distribución de sistema operativo
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 2
Prioridad en el negocio: Media	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Medio	**Puntos reales:** 1 Semana
Descripción: Las distribuciones incluyen el nombre y la descripción. Se gestionan en el sistema, permitiendo *insertar*, *editar*, *eliminar* y *mostrar* mediante filtros cada uno de ellas.	
Observaciones: En la interfaz web de SINPM: 1. Para mostrar mediante filtros se selecciona en el menú el enlace *Distribuciones*, submenú de *Sistemas Operativos* y se muestra un listado permitiendo filtrar mediante: nombre.	

2. Para insertar una distribución de un sistema operativo son necesarios los siguientes datos: nombre y descripción. • Si existe previamente, se muestra un mensaje de error. 3. Para editar la información de una distribución, se selecciona, se modifican los datos y posteriormente se actualiza el elemento. 4. Para eliminar una distribución, se selecciona previamente y se confirma la eliminación. • Si está relacionada con un sistema operativo, no se elimina y se muestra un mensaje de error.
Prototipo de interfaz: Figura A.9 (ver anexos)

Tabla 2.6: HU: Gestionar distribución de sistema operativo

Gestionar arquitectura de sistema operativo

Historia de Usuario	
Número: 5	**Nombre de Historia de Usuario:** Gestionar arquitectura de sistema operativo
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 2
Prioridad en el negocio: Media	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Medio	**Puntos reales:** 1 Semana
Descripción: Las arquitecturas incluyen el nombre y la descripción. Se gestionan en el sistema, permitiendo *insertar*, *editar*, *eliminar* y *mostrar* mediante filtros cada una de ellas.	
Observaciones: En la interfaz web de SINPM: 1. Para mostrar mediante filtros se selecciona en el menú el enlace *Arquitectura*, submenú de *Sistemas Operativos*, y se muestra un listado permitiendo filtrar mediante: nombre. 2. Para insertar arquitectura de un sistema operativo son necesarios los siguientes datos: nombre y descripción. • Si existe previamente, se muestra un mensaje de error. 3. Para editar la información de una arquitectura, se selecciona, se modifican los datos y posteriormente se actualiza el elemento. 4. Para eliminar una arquitectura, se selecciona previamente y se confirma la eliminación. • Si está relacionada con un sistema operativo, no se elimina y se muestra un error.	
Prototipo de interfaz: Figura A.10 (ver anexos)	

Tabla 2.7: HU: Gestionar arquitectura de sistema operativo

Gestionar sistema operativo

Historia de Usuario	
Número: 6	**Nombre de Historia de Usuario:** Gestionar sistema operativo
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 2 Semanas
Riesgo en el desarrollo: Alto	**Puntos reales:** 1 Semana
Descripción: Los sistemas operativos contienen características de interés como lo son: la arquitectura (tabla 2.7), la distribución a la que pertenecen (tabla 2.6), la variante, la versión y la plataforma. Se gestionan en el sistema, permitiendo *insertar*, *editar*, *eliminar* y *mostrar* mediante filtros cada uno de ellos.	
Observaciones: En la interfaz web de SINPM: 1. Para mostrar mediante filtros se selecciona en el menú el enlace *Sistemas Operativos* y se muestra un listado permitiendo filtrar mediante: versión, variante, arquitectura y plataforma. 2. Para insertar un sistema operativo son necesarios los siguientes datos: nombre de la distribución, versión, variante, arquitectura y plataforma. • Si existe previamente, se muestra un mensaje de error. 3. Para editar la información de un sistema operativo, se selecciona, se modifican los datos y posteriormente se actualiza el elemento. 4. Para eliminar un sistema operativo, se selecciona previamente y se confirma la eliminación. • Si está relacionado con una certificación, no se elimina y se muestra un error.	
Prototipo de interfaz: Figura A.11 (ver anexos)	

Tabla 2.8: HU: Gestionar sistema operativo

Gestionar driver

Historia de Usuario	
Número: 7	**Nombre de Historia de Usuario:** Gestionar driver
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Alto	**Puntos reales:** 1 Semana
Descripción: Los drivers incluyen el nombre, versión y dos direcciones web para descargar donde una es opcional. Se gestionan en el sistema, permitiendo *insertar*, *editar*, *eliminar* y *mostrar* mediante filtros cada uno de ellos.	
Observaciones: En la interfaz web de SINPM: 1. Para mostrar mediante filtros se selecciona en el menú el enlace *Controladores* y se muestra un listado permitiendo filtrar mediante: nombre, versión y dirección url principal. 2. Para insertar un driver son necesarios los siguientes datos: nombre, versión, dirección web oficial y opcionalmente otra dirección web donde se pueda encontrar. • Si existe previamente, se muestra un mensaje de error. 3. Para editar un driver, se selecciona, se modifican los datos y posteriormente se actualiza el elemento. 4. Para eliminar un driver, se selecciona previamente y se confirma la eliminación. • Si está relacionado con una certificación, no se elimina y se muestra un error.	
Prototipo de interfaz: Figura A.12 (ver anexos)	

Tabla 2.9: HU: Gestionar driver

Gestionar categorías de certificados de hardware

Historia de Usuario	
Número: 8	**Nombre de Historia de Usuario:** Gestionar categoría de certificación
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 1

Prioridad en el negocio: Alta	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Alto	**Puntos reales:** 1 Semana
Descripción: Una categoría de certificación incluye el nombre, la descripción, un logo de la certificación opcionalmente, así como la distribución a la que pertenece. Se gestionan en el sistema, permitiendo *insertar*, *editar*, *eliminar* y *mostrar* mediante filtros cada una de ellas.	
Observaciones: En la interfaz web de SINPM: 1. Para mostrar mediante filtros se selecciona en el menú el enlace *Categorías*, submenú de *Certificaciones* y se muestra un listado permitiendo filtrar mediante: nombre y sistema operativo al que se aplica la categoría de certificación. 2. Para insertar un categoría de certificación son necesarios los siguientes datos: nombre, descripción, la distribución a la que pertenece y opcionalmente una imagen que represente a la misma. • Si existe previamente, se muestra un mensaje de error. 3. Para editar la información de una categoría de certificación, se selecciona, se modifican los datos y posteriormente se actualiza el elemento. 4. Para eliminar una categoría de certificación, se selecciona previamente y se confirma la eliminación. • Si está relacionada con un certificado, no se elimina y se muestra un error.	
Prototipo de interfaz: Figura A.13 (ver anexos)	

Tabla 2.10: HU: Gestionar categoría de certificación

Gestionar certificados de hardware

Historia de Usuario	
Número: 9	**Nombre de Historia de Usuario:** Gestionar certificados de hardware
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 2 Semanas
Riesgo en el desarrollo: Alto	**Puntos reales:** 1 Semana

Descripción: La certificación de compatibilidad define si el hardware inventariado es compatible con determinado sistema operativo, teniendo en cuenta que el controlador define esta relación, pues él hace compatible o no al hardware con el sistema operativo. Para realizar la certificación es necesario establecer la información siguiente: categoría de certificación que se emplea, el usuario que creó el certificado, el driver con el que se comprobó la compatibilidad, la descripción, un documento predefinido en la institución que formalice la certificación y otro documento que contiene en caso de necesitarlo una configuración probada para el hardware. Se gestionan los certificados permitiendo *insertar*, *editar*, *eliminar* y *mostrar* mediante filtros cada uno ellos.
Observaciones: En la interfaz web de SINPM: 1. Para mostrar mediante filtros se selecciona en el menú el enlace *Certificaciones* y se muestra un listado permitiendo filtrar mediante: fabricante del hardware, modelo del hardware, versión del hardware y categoría de la certificación. 2. Para mostrar un certificado de hardware, se selecciona el certificado y se muestra toda su información: descripción, documentos que lo acreditan, así como la configuración probada si está establecida, el hardware, el sistema operativo y el controlador al que está relacionado. 3. Para insertar un certificado de hardware, es necesario tabular la información referente a la descripción, seleccionar el hardware, y el sistema operativo. Además es necesario seleccionar de los drivers que están relacionados con el hardware, cuál es el que se utilizó para la certificación. Se debe seleccionar la categoría de certificación y adjuntar un documento que acredita la misma. Opcionalmente se puede adjuntar otro documento con las configuraciones probadas del hardware. 4. Para editar la información de una certificación, se selecciona el elemento, se modifican sus campos y se actualiza posteriormente. 5. Para eliminar una certificación, se selecciona y se confirma la eliminación.
Prototipo de interfaz: Figura A.14 (ver anexos)

Tabla 2.11: HU: Gestionar certificados de hardware

Gestionar reglas de importación de hardware

Historia de Usuario	
Número: 10	**Nombre de Historia de Usuario:** Gestionar reglas de importación de hardware
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 2
Prioridad en el negocio: Media	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Medio	**Puntos reales:** 1 Semana
Descripción: Las reglas de importación incluyen el nombre del sistema remoto desde el cual se importa, el nombre de la categoría con la que se relaciona a este sistema y el nombre de la categoría que presenta. Al importar hardware, si este no presenta una categoría homóloga a este sistema se le ignora. Las reglas se gestionan en el sistema, permitiendo *insertar*, *editar*, *eliminar* y *mostrar* mediante filtros cada una de ellas.	
Observaciones: En la interfaz web de SINPM: 1. Para mostrar mediante filtros se selecciona en el menú el enlace *Reglas* y se muestra un listado permitiendo filtrar mediante: sistema de importación, nombre de categoría desde la que se importa y categoría a la que se importa. 2. Para insertar una regla de importación son necesarios los siguientes datos: nombre del sistema remoto, nombre de la categoría de este sistema a la cual se hace referencia y el nombre de la categoría foránea que se importa. • Si existe previamente, se muestra un mensaje de error. 3. Para editar la información de una regla de importación, se selecciona, se modifican los datos y posteriormente se actualiza el elemento. 4. Para eliminar una regla de importación, se selecciona previamente y se confirma la eliminación.	
Prototipo de interfaz: Figura A.15 (ver anexos)	

Tabla 2.12: HU: Gestionar reglas de importación de hardware

Buscar hardware

Historia de Usuario	
Número: 11	**Nombre de Historia de Usuario:** Buscar hardware
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Alto	**Puntos reales:** 1 Semana
Descripción: Permite encontrar rápidamente un hardware mediante la búsqueda por condicionales y operadores lógicos.	
Observaciones: En la interfaz web de SINPM: 1. Para buscar hardware, son necesarios algunos de los siguientes campos separados por espacios, usando comodines y operadores lógicos de búsqueda: fabricante, modelo y versión	
Prototipo de interfaz: Figura A.16 (ver anexos)	

Tabla 2.13: HU: Buscar hardware

Importar reporte de hardware

Historia de Usuario	
Número: 12	**Nombre de Historia de Usuario:** Importar reporte de hardware
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 2
Prioridad en el negocio: Alta	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Alto	**Puntos reales:** 1 Semana
Descripción: Permite importar un conjunto de hardware por cualquier sistema que esté autenticado con el sistema de seguridad y tenga habilitado el permiso para este servicio. Excepcionalmente el sistema SistDCH sólo necesita un token de seguridad que este sistema define en la configuración inicial.	
Observaciones: En la interfaz web de SINPM:	

1. Para importar el reporte se debe enviar un objeto SOAP con la estructura como se muestra en el código A.2 (ver anexos), como resultado se devuelven los identificadores de cada dispositivo insertado, en caso de no poderse insertar por faltar datos al hardware como: categoría, nombre del fabricante o modelo; el identificador devuelto es un uno negativo. • El hardware que estaba previamente no es modificado, sólo se actualizan las relaciones de compatibilidad con los sistemas operativos y los drivers que usa.
Prototipo de interfaz: no tiene, es a través de servicios web

Tabla 2.14: HU: Importar reporte de hardware

Importar reporte de hardware en XML desde SistDCH

Historia de Usuario	
Número: 13	**Nombre de Historia de Usuario:** Importar reporte de hardware en XML desde SistDCH
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 2
Prioridad en el negocio: Media	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Medio	**Puntos reales:** 1 Semana
Descripción: Permite agregar una lista de hardware analizado y exportado a XML por SistDCH.	
Observaciones: En la interfaz web de SINPM: 1. Para importar el reporte desde un fichero generado por SistDCH, es necesario adjuntar el fichero que genera la herramienta. • El hardware que estaba previamente no es modificado, sólo las relaciones de compatibilidad y los drivers. Posteriormente de la importación satisfactoria se muestra la cantidad de hardware nuevo. • Si el fichero contiene errores, se muestra un mensaje de error.	
Prototipo de interfaz: Figura A.17 (ver anexos)	

Tabla 2.15: HU: Importar reporte de hardware en XML desde SistDCH

Buscar hardware homólogo

Historia de Usuario	
Número: 14	**Nombre de Historia de Usuario:** Buscar hardware homólogo
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 2 Semanas
Riesgo en el desarrollo: Alto	**Puntos reales:** 1 Semana
Descripción: La búsqueda de hardware homólogo es una de las funcionalidades más importantes, se basa en buscar hardware similar que esté certificado. Mientras más datos técnicos sean iguales mayor cercanía tendrá la búsqueda de encontrar el hardware homólogo.	
Observaciones: En la interfaz web de SINPM: 1. Para buscar hardware homólogo, primero se busca como se muestra en la tabla 2.13, posteriormente se selecciona la opción homologar y se selecciona para que categoría de certificación se desea buscar homólogos. Al finalizar la búsqueda se muestra la lista de hardware homologado.	
Prototipo de interfaz: Figura A.18 (ver anexos)	

Tabla 2.16: HU: Buscar hardware homólogo

Gestionar notificaciones

Historia de Usuario	
Número: 15	**Nombre de Historia de Usuario:** Gestionar Notificaciones
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 3

Prioridad en el negocio: Baja	**Puntos estimados:** 1 Semanas
Riesgo en el desarrollo: Bajo	**Puntos reales:** 1 Semana
Descripción: Permite gestionar las notificaciones, las cuales se generan automáticamente cuando se efectúan acciones que son importantes para el administrador del sistema como la importación de hardware nuevo y la eliminación de certificaciones de hardware. Se gestionan las notificaciones permitiendo *Mostrar*, *Listar* y *Eliminar* estas.	
Observaciones: En la interfaz web de la SINPM: 1. Para Listar notificaciones se selecciona la acción Notificaciones en el menú 2. Para Eliminar notificaciones se selecciona y se ejecuta la acción eliminar.	
Prototipo de interfaz: Figura A.19 (ver anexos)	

Tabla 2.17: HU: Gestionar Notificaciones

Gestionar comentarios de compatibilidad de hardware

Historia de Usuario	
Número: 16	**Nombre de Historia de Usuario:** Gestionar comentarios de compatibilidad de hardware
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 3
Prioridad en el negocio: Baja	**Puntos estimados:** 1.5 Semanas
Riesgo en el desarrollo: Bajo	**Puntos reales:** 1 Semana
Descripción: Permite gestionar los comentarios de la compatibilidad de hardware que hacen los usuarios. Se gestionan los comentarios permitiendo *comentar*, *listar*, *editar*, y *eliminar* estos.	
Observaciones: En la interfaz web de la SINPM: 1. Para comentar la compatibilidad de un hardware en un sistema operativo,el usuario debe seleccionar el hardware para mostrar su información, posteriormente seleccionar la opción comentar dentro del listado de compatibilidad del hardware y agregar la información.	

2. Para listar comentarios a Hardware se selecciona el hardware y se listan al final de la descripción del mismo. 3. Para eliminar un comentario de hardware, se busca el comentario en el listado de comentarios y se selecciona eliminar.
Prototipo de interfaz: Figura A.20 (ver anexos)

Tabla 2.18: HU: Gestionar comentarios de compatibilidad de hardware

Evaluar compatibilidad de hardware

Historia de Usuario	
Número: 17	**Nombre de Historia de Usuario:** Evaluar compatibilidad hardware
Modificación: 1	
Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 3
Prioridad en el negocio: Baja	**Puntos estimados:** 1.5 Semanas
Riesgo en el desarrollo: Bajo	**Puntos reales:** 1 Semana
Descripción: Permite evaluar el grado de compatibilidad del hardware según el usuario.	
Observaciones: En la interfaz web de SINPM: 1. Para evaluar la compatibilidad de un hardware con un sistema operativo, se muestra el mismo y en el listado de compatibilidad se establece una puntuación en base a 5 puntos. 2. Para mostrar la evaluación de la compatibilidad se selecciona el hardware y en el listado de compatibilidad del mismo se muestra la cantidad de usuarios que han votado, así como la valoración general.	
Prototipo de interfaz: Figura A.21 (ver anexos)	

Tabla 2.19: HU: Evaluar compatibilidad de hardware

Descargar SistDCH

Historia de Usuario	
Número: 18	**Nombre de Historia de Usuario:** Descargar SistDCH
Modificación: 1	

Usuario: Ricardo Ramón Carbajo Pérez	**Iteración asignada:** 3
Prioridad en el negocio: Baja	**Puntos estimados:** 1 Semana
Riesgo en el desarrollo: Bajo	**Puntos reales:** 1 Semana
Descripción: Permite descargar la aplicación cliente para reportar el hardware compatible del usuario.	
Observaciones: En la interfaz web de SINPM: 1. Para descargar SistDCH, se selecciona el enlace a la aplicación que se encuentra en la vista principal del menú *Catálogo*, y posteriormente comenzará la descarga donde el usuario seleccione.	
Prototipo de interfaz: Figura A.22 (ver anexos)	

Tabla 2.20: HU: Descargar SistDCH

2.5 Diseño del Sistema de Certificación y Homologación de Hardware

2.5.1 Modelo del diseño

El modelo de diseño proporciona detalles acerca de las estructuras de datos, las arquitecturas, las interfaces y los componentes del software que son necesarios para implementar el sistema. Un modelo de diseño representa los requisitos funcionales en un lenguaje técnico lo más cercano posible al lenguaje de programación y sirve como una abstracción del código fuente.

El diseño de SICEH, como parte de la PMSWL, debe acoplarse a la arquitectura de la misma (SOA + Presentación desacoplada), por lo que se implementa como una aplicación mediante el framework Symfony, en el cual se define una capa de servicios para la integración con la plataforma y se implementa un módulo de interfaz de usuario en el sistema SINPM como se puede apreciar la siguiente figura:

Para llevar a cabo el diseño se utilizaron los siguientes estilos arquitectónicos [2]:

[2]Un estilo describe una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Esas piezas se encuentran repetidamente en la práctica, trasuntando la existencia de decisiones estructurales coherentes.[18]

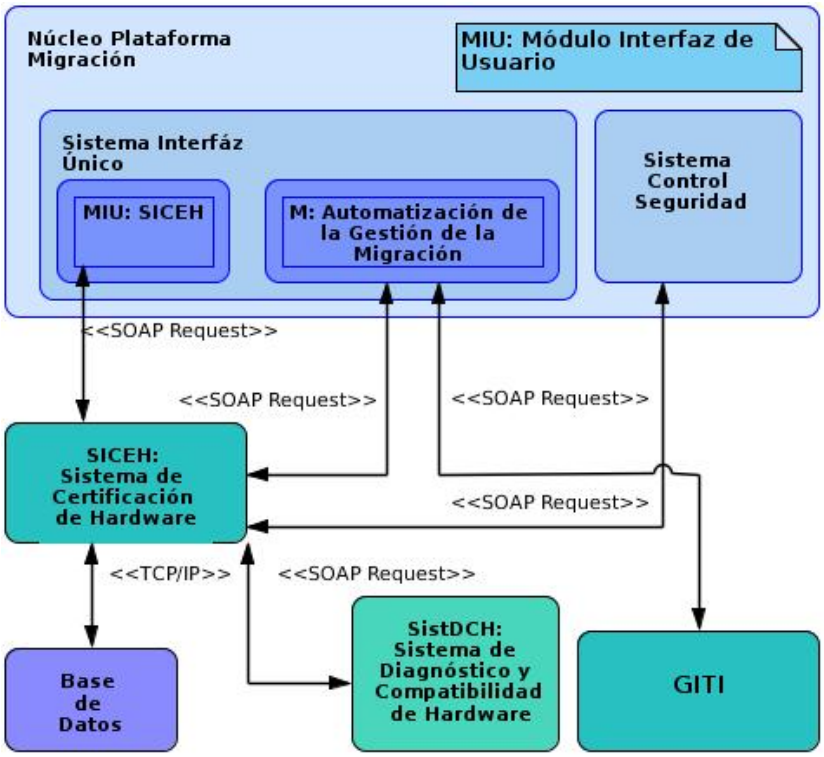

Figura 2.2: Diagrama de la Vista Arquitectónica

SOA Es una arquitectura que utiliza como núcleo conceptual proveedores de servicios y consumidores de servicios [1], aplicando varias tecnologías como xml para el estructurar los mensajes, wsdl como directorio de servicios y SOAP como protocolo de comunicación. En esta arquitectura los sistemas clientes o consumidores dependen de la publicación de los servicios como se puede apreciar en la fig. A.2 en los anexos. El uso de la arquitectura SOA en la PMSWL permite que pueda funcionar con sus sistemas segregados o centralizados, en dependencia del entorno en el que se despliegue la misma, y los recursos que existan para ello.

Arquitectura en Capas Una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior [7]. Para lograr la integración con la PMSWL utilizando la arquitectura SOA se deben publicar los servicios por lo que se debe exponer la lógica de negocio mediante interfaces de servicios. Lo que permite modelar el sistema en varias capas como son la Capa de Presentación, la Capa del Dominio y la Capa de Acceso a Datos como se muestra en la fig. A.23 en los anexos.

En el diagrama de clases del diseño en la fig. A.23 (ver anexos) puede apreciarse que se hace una separación lógica en

tres capas, siguiendo un estilo N-Capas como se expresa en la definición anterior (sección 2.5.1). La capa presentación contiene la clase ckSOAPHandler cuya función es manejar las peticiones SOAP como controlador frontal, traduciendo las funcionalidades en servicios web. En la capa dominio se encuentran las clases que controlan la lógica del negocio, donde se encuentra las clases controladoras actions.php que encapsulan funcionalidades de las HU explicadas con anterioridad. La relación de las clases controladoras con las clases de la capa acceso a datos, permiten abstraer el sistema gestor de bases de datos que se utilice.

Patrones de diseño utilizados

El framework Symfony implementa una serie de patrones de diseño que hacen su arquitectura sea suficientemente robusta y a la vez flexible como para adaptarse a los casos más complejos. El marco de trabajo mencionado es capaz de fusionar buenas prácticas de trabajo por sí mismo, de forma que los desarrolladores no tengan que preocuparse por implementar varios de los patrones arquitectónicos y de diseño más utilizados en la actualidad, ya que el mismo los implementa.

Patrones GRASP

Creador Todos los módulos del sistema tienen una clase actions.class.php que contiene las acciones definidas para dichos módulos y es en ella misma donde se ejecutan las funciones que hacen al sistema funcional. En esta clase las acciones se encargan de crear los objetos de las clases que representan las entidades, evidenciando de este modo que la clase actions.class.php es el "creador" de las entidades.

Controlador Frontal Es un patrón de diseño web usado como único punto de entrada a la aplicación. Realiza tareas comunes a todos los controladores, manejo de la seguridad, de las peticiones de los usuarios, carga la configuración de la aplicación y delega la responsabilidad de responder a las peticiones al módulo específico que tiene la acción enviada en la petición.

Alta Cohesión Symfony permite la asignación de responsabilidades con alta cohesión, la clase actions.class.php tiene la responsabilidad para definir las acciones sobre las plantillas y colabora con otras para realizar diferentes operaciones. Proporcionando que el software sea flexible frente a grandes cambios.

Experto Es un patrón de asignación de responsabilidades, que delega la responsabilidad a quien contiene la información necesaria para cumplirla. Tal es el caso de las clases pertenecientes a la capa del modelo (capa integrante del estilo arquitectónico Modelo-Vista-Controlador) que genera el ORM Doctrine con todos los atributos y métodos útiles, son las encargadas de acceder a la base de datos y devolver los mismos de acuerdo con la información que ellas manipulan.

Patrones GOF

Singleton (Única instancia) Encapsula la creación de objetos con distintas familias de implementaciones. Es un patrón creacional que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. El patrón se utiliza dentro del framework y en la mayoría de sus plugins, pues permite reutilizar una instancia de un objeto que se hace persistente durante la ejecución del framework. A través del método estático sfContext::getInstance(), se obtiene la referencia del objeto que engloba a todos los objetos del núcleo.

Fábrica Abstracta Se utiliza este patrón al trabajar con objetos de distintas familias; de manera que no se mezclen entre sí, haciendo transparente el tipo de familia concreta que se esté usando. Cuando el marco de trabajo necesita, por ejemplo, crear un nuevo objeto, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

2.5.2 Integración con sistemas externos

La integración de SICEH con sistemas externos a través de servicios web, es asegurada mediante el uso de cabeceras SOAP mediante un token de seguridad de un usuario registrado en el SCS. El token es el identificador de sesión del usuario en la PMSWL. Las categorías de SICEH son flexibles a cualquier orden, por tanto las categorías que use el sistema externo, deben ser asociadas mediante reglas dentro del módulo de certificación. Las peticiones de importación con categorías no relacionadas, se toman como datos irrelevantes y no se agregan al sistema. En el caso de SistDCH no es necesario un usuario del SCS pues sólo necesita un token de seguridad que se define en la configuración de SICEH.

1. A partir de la comunicación confiable que permite SCS, el sistema externo debe inicialmente buscar el identificador

del sistema operativo en SICEH mediante el servicio: **int** getOperativeSystemIdByUid(string $uid), en el cual se pasa como parámetro una cadena separada por espacios o guiones, que hace referencia a los datos del sistema operativo de la forma: [Distribución] [Versión] [Arquitectura[3]] [Plataforma[4]] [Variante[5]].

2. El servicio antes descrito devuelve el identificador del sistema operativo, lo que permite comenzar la importación mediante el servicio **int** [] importHardware(HardwareReportMeta $report, string $operativSystem), el cual devuelve un listado de los identificadores del hardware importado, si no se pudo insertar alguno se devuelve el número -1 como valor. El identificador del sistema operativo es opcional siempre que dentro de la clase HardwareReport (véase código A.3 en anexos) se defina el mismo para cada objeto de la lista en la clase HardwareReportMeta.

2.6 Diseño del módulo de interfaz de usuario

2.6.1 Modelo del diseño

El diseño del módulo está provisto por la arquitectura de SINPM [27]. Se basa en una estructura modular orientada a consumir los servicios de SICEH, a continuación se describen los estilos arquitectónicos empleados:

Presentación Desacoplada PMSWL utiliza este estilo arquitectónico para simplificar el código de las interfaces y para que la plataforma pueda disponer de varias interfaces para una misma lógica de negocio, aumentando así la interoperabilidad. Es un requisito necesario por tanto se aplica este estilo en el módulo.

Modelo-Vista-Controlador (MVC) Separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

En el diagrama de clases del diseño en la fig. A.27 (ver anexos) puede apreciarse que la arquitectura de la PMSWL separa la estructura del módulo en tres capas mediante el estilo MVC descrito anteriormente. Inicialmente se genera un documento html mediante el fichero index.php en el navegador, la clase Enrutador es la responsable de cargar las configuraciones, los controladores de las acciones y las demás clases. Inicialmente se carga el módulo por defecto, el

[3]La arquitectura de computadoras es el diseño conceptual y la estructura operacional fundamental de un sistema de computadora. Dentro de las arquitecturas más usadas se encuentra i386 y AMD.

[4]Se refiere a bus de datos del hardware el cual puede ser de 32 o 64 bits.

[5]Variación que tienen las distribuciones en cuanto su objetivo, ya sea para servidor, escritorio o cualquier otra denominación.

contenido es generado mediante peticiones AJAX hacia el fichero *contenido.php* el cual retorna el módulo y el menú en dos peticiones. En la fig. A.26 (ver anexos) se puede ver el diseño de clases simplificado para la gestión de categorías.

La clase **CoreServicio** gestiona el acceso a los servicios web, lo que permite la comunicación con SICEH, para ello provee automáticamente las credenciales necesarias, controla los errores que generen los servicios estableciendo un estándar para acceder a los resultados de los mismos. **CoreAccion** provee una interfaz única para la ejecución de las diferentes acciones de los módulos. **CoreContext** tiene funcionalidades comúnmente usadas y que dependen del contexto. **CorePaginar** provee utilidades para el paginado de las vistas en las que se lista información.

La clase **Action** es la aplicación del patrón de diseño *adapter* debido a que como ya existen varios módulos que usan la clase CoreAccion, se extendió la misma definiéndola como una generalización. En esta clase mediante el constructor se crea un objeto CoreServicio como atributo de la clase y se redefine la configuración de la clase SOAPClient perteneciente a CoreAction para mapear los objetos que se van a intercambiar con SICEH mediante el protocolo SOAP.

2.7 Conclusiones

En este capítulo se definen las funcionalidades que el sistema debe cumplir desglosados en historias de usuario y estableciéndose un orden de prioridad para su realización. Se describe el diseño del sistema y el módulo de interfaz en la plataforma de migración por separado, así como la arquitectura y los patrones de diseño que se usan.

3

Implementación y prueba del sistema

3.1 Introducción

Este capítulo aborda la implementación del sistema haciendo énfasis en las pruebas, así como la planificación de las iteraciones del ciclo de desarrollo. Además se define el impacto y aporte de la solución propuesta.

3.2 Implementación

3.2.1 Planificación

Durante el proceso de desarrollo de software un elemento vital es la planificación, esta proporciona un marco de trabajo que permita estimar razonablemente recursos, costo y planificación temporal. En la metodología utilizada la planificación tiene como propósito establecer la visión, fijar expectativas, y asegurar financiamiento del producto. Define cuáles son las historias de usuario más significativas, y las ubican en las iteraciones según esta prioridad. Divide el proceso de desarrollo de software en iteraciones, planificando el trabajo a realizar en cada una de ellas.

Plan de releases

De acuerdo a las funcionalidades descritas en las historias de usuarios y en concordancia con la prioridad asignada para su realización, se planificaron cuatro iteraciones que recogen el desarrollo del SICEH y del módulo.

Release	Descripción	Orden de la HU a Implementar	Duración
1	Desarrollo de las Historias de Usuario de alta prioridad	3,6,7,8,11,16	10 semanas
2	Desarrollo de las Historias de Usuario de media prioridad	1,2,4,5,9,10,12,13,14,15,16	5 semanas
3	Desarrollo de las Historias de Usuario de baja prioridad	17,18,19,20	4 semanas
4	Integrar el producto con el núcleo de la Plataforma		1 Semana

Tabla 3.1: Plan de Releases

3.2.2 Estándar de codificación utilizado

Una de las buenas prácticas propuesta por las metodologías ágiles, es la adopción de un estándar de codificación por parte del equipo de desarrollo; asegurando que el código exprese claramente el propósito del mismo y agilice el proceso de refactorización. Para darle cumplimiento a la solución propuesta se utilizó el estándar de código definido por el equipo de desarrollo de la plataforma de migración.

3.2.3 Estructura de componentes

Los componentes o ficheros pertenecientes a SICEH mostrado en el diagrama de componentes en la fig. A.24 (ver anexos), mantienen una organización física estrechamente ligada a la estructura lógica de Symfony mostrada en la fig. A.28 (ver anexos), dentro de la carpeta apps donde se encuentra la aplicación en este caso denominada frontend. Existe una carpeta dedicada a los módulos (modules), en la cual se encuentran los controladores encargados de la lógica del negocio. Gracias a la granularidad de las configuraciones de Symfony en la aplicación, existe una carpeta para las configuraciones de la aplicación (config), en la cual se definen variables específicas de la misma (véase, fig. A.29 en anexos) que permiten la integración con la PMSWL. En la carpeta lib se encuentran las clases que representan el modelo, las mismas son empleadas en el intercambio de datos mediante SOAP. En la carpeta que representa al framework, se exponen los dos ficheros fundamentales el schema.yml donde se define la estructura de la base de datos y

un subpaquete ckWebServicePlugin que representa al plugin que maneja las peticiones SOAP. El fichero cert_hwWS.php es el controlador frontal de las peticiones SOAP.

3.2.4 Distribución física del sistema

En la fig. 3.1 puede apreciarse el diagrama de despliegue en el que se muestran tres servidores de aplicaciones donde se encuentran los sistemas proveedores de servicios: SICEH y SCS. En el sistema de interfaces SINPM se encuentran los módulos de aplicación de la PMSWL.

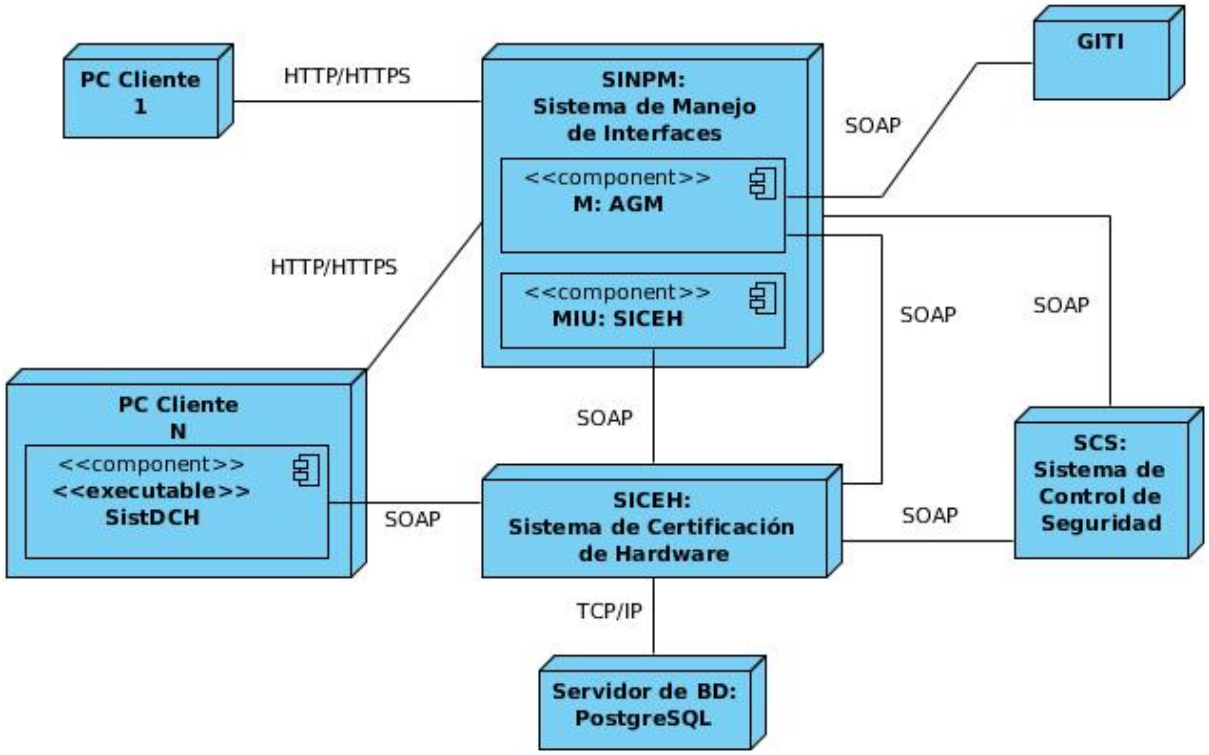

Figura 3.1: Diagrama de Despliegue

En la figura anterior se representan los sistemas mediante un modelo distribuido, sin embargo, los tres sistemas pueden estar alojados en el mismo servidor gracias a la flexibilidad que brinda la arquitectura SOA. El software cliente SistDCH como es una unidad de software o componente [8], se comunica con SICEH enviando los datos técnicos del hardware. El

módulo MAGM, es el encargado de automatizar el proceso de migración y en este caso relaciona todo el hardware recolectado por GITI al sistema de certificación propuesto. SICEH está provisto por una base de datos la cual por definición debe ser gestionada por PostgreSQL, aunque Symfony admite una gran gama de alternativas.

3.3 Pruebas

Las pruebas en el desarrollo de software es un proceso vital, en la programación extrema define entre iteración e iteración, casos de pruebas para poder avanzar a una iteración superior. Durante el desarrollo del SICEH se diseñaron un conjunto de casos de prueba a las que fue sometido el sistema para comprobar el funcionamiento de acuerdo con las historias de usuario.

Para la realización de las pruebas se debe tener en cuenta que:

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

Dentro de SXP se establecen dos tipos de pruebas, las unitarias la cuales se escriben directamente para el código y las elabora el desarrollador antes de escribir el código y las pruebas funcionales que las hace el cliente para las historias de usuario.

3.3.1 Pruebas funcionales

> "Las pruebas funcionales no sólo validan la transformación de una entrada en una salida, sino que validan una característica completa ... validan un proceso y requieren un escenario" [29].

Este tipo de pruebas tienen un gran valor, pues validan si las funcionalidades están implementadas correctamente y de no estarlo corregir los errores encontrados. Symfony integra el framework de pruebas Lime, que convierte un proceso manual e iterativo en uno automatizado y sencillo de realizar. Permite aplicar las pruebas y obtener los resultados en todo momento mostrando un resumen de las mismas.

Para probar los servicios web se utiliza una clase que se encuentra integrada al plugin ckWebServicePlugin, denominada **ckTestSoapClient** la cual permite hacer las comprobaciones mediante peticiones SOAP. Las pruebas son revisadas y aprobadas por el cliente después de ejecutadas. En el código A.1 (ver anexos) se muestra un ejemplo de una prueba utilizando lime y la clase del plugin.

Para cada módulo del sistema se implementaron pruebas funcionales, las cuales se ejecutan en SICEH mediante el comando: >**php** symfony test: functional frontend

A continuación se relacionan las pruebas de las historias de usuario con mayor impacto:

Caso de Prueba: HU Gestionar hardware

Caso de Prueba	
Código: PMSWL-SICEH-3	**Nombre Historia de Usuario:** Gestionar hardware
Nombre de la Persona que realiza la Prueba: Ricardo Ramón Carbajo Pérez	
Descripción de la prueba: Conjunto de pruebas de inserción, actualización, eliminación y listado de hardware.	
Condiciones de ejecución: La prueba se realiza mediante la shell del sistema operativo.	
Entrada / Pasos de ejecución: Dentro de la raíz del directorio donde se encuentra la aplicación, se ejecuta el comando: *php symfony test:functional frontend hu_3*	
Resultado esperado: La salida debe mostrar que no existen errores en las pruebas.	
Evaluación de la prueba: Satisfactorio.	
Resultado esperado: La salida debe mostrar que no existen errores en las pruebas.	
Figura A.30 (ver anexos)	

Tabla 3.2: CP: HU Gestionar hardware

Caso de Prueba: HU Gestionar sistema operativo

Caso de Prueba	
Código: PMSWL-SICEH-6	**Nombre Historia de Usuario:** Gestionar Sistema Operativo
Nombre de la Persona que realiza la Prueba: Ricardo Ramón Carbajo Pérez	

Descripción de la Prueba: Conjunto de pruebas de inserción, actualización, eliminación y listado de hardware.
Condiciones de Ejecución: La prueba se realiza mediante la shell del sistema operativo.
Entrada / Pasos de ejecución: Dentro de la raíz del directorio donde se encuentra la aplicación, se ejecuta el comando: *php symfony test:functional frontend hu_6*
Resultado esperado: La salida debe mostrar que no existen errores en las pruebas.
Evaluación de la prueba: Satisfactorio
Resultado esperado: La salida debe mostrar que no existen errores en las pruebas.
Figura A.31 (ver anexos)

Tabla 3.3: CP: HU Gestionar sistema operativo

Caso de Prueba: HU Gestionar certificación

Caso de Prueba	
Código: PMSWL-SICEH-9	**Nombre Historia de Usuario:** Gestionar certificación
Nombre de la Persona que realiza la Prueba: Ricardo Ramón Carbajo Pérez	
Descripción de la prueba: Conjunto de pruebas de inserción, actualización, eliminación y listado de hardware. Para la inserción se utilizaron un conjunto de datos de 100 dispositivos de impresoras.	
Condiciones de ejecución: La prueba se realiza mediante la shell del sistema operativo.	
Entrada / Pasos de ejecución: Dentro de la raíz del directorio donde se encuentra la aplicación, se ejecuta el comando *php symfony test:functional frontend hu_9*	
Resultado esperado: La salida debe mostrar que no existen errores en las pruebas.	
Evaluación de la prueba: Satisfactorio	
Resultado esperado: La salida debe mostrar que no existen errores en las pruebas.	
Figura A.32 (ver anexos)	

Tabla 3.4: CP: HU Gestionar certificación

Caso de Prueba: HU Importar hardware

Caso de Prueba	
Código: PMSWL-SICEH-12	**Nombre Historia de Usuario:** Importar hardware

Nombre de la Persona que realiza la Prueba: Ricardo Ramón Carbajo Pérez
Descripción de la prueba: Conjunto de pruebas de inserción, actualización, eliminación y listado de hardware. Para la inserción se utilizaron un conjunto de datos de 100 dispositivos de impresoras.
Condiciones de ejecución: La prueba se realiza mediante la shell del sistema operativo.
Entrada / Pasos de ejecución: Dentro de la raíz del directorio donde se encuentra la aplicación, se ejecuta el comando *php symfony test:functional frontend hu_12*
Resultado esperado: La salida debe mostrar que no existen errores en las pruebas.
Evaluación de la prueba: Satisfactorio
Resultado esperado: La salida debe mostrar que no existen errores en las pruebas.
Figura A.33 (ver anexos)

Tabla 3.5: CP: HU Importar hardware

Caso de Prueba: HU Buscar hardware homólogo

Caso de Prueba	
Código: PMSWL-SICEH-14	**Nombre Historia de Usuario:** Buscar hardware homólogo
Nombre de la Persona que realiza la Prueba: Ricardo Ramón Carbajo Pérez	
Descripción de la prueba: Conjunto de pruebas de inserción, actualización, eliminación y listado de hardware. Para la inserción se utilizaron un conjunto de datos de 100 dispositivos de impresoras.	
Condiciones de ejecución: La prueba se realiza mediante la shell del sistema operativo.	
Entrada / Pasos de ejecución: Dentro de la raíz del directorio donde se encuentra la aplicación, se ejecuta el comando *php symfony test:functional frontend hu_12*	
Resultado esperado: La salida debe mostrar que no existen errores en las pruebas.	
Evaluación de la prueba: Satisfactorio.	
Resultado esperado: La salida debe mostrar que no existen errores en las pruebas.	
Figura A.34 (ver anexos)	

Tabla 3.6: CP: HU Buscar hardware homólogo

3.4 Impacto y aporte de la solución propuesta

El mayor impacto del sistema desarrollado reside en el comienzo de la migración de las empresas a SWL, define en conjunto con el MAGM todo el hardware compatible, lo que permite determinar si es necesario buscar alternativas para mantener la migración. Como resultado se obtiene una aplicación web libre que automatiza el proceso de certificación y homologación de hardware de una forma efectiva, flexible y adaptable ante cambios. El sistema permite la gestión de hardware y sus especificaciones técnicas, sus relaciones de compatibilidad con los sistemas operativos, así como las certificaciones y valoraciones de los usuarios. El uso de la aplicación obtenida hace posible que el trabajo de los especialistas en migración sea más rápido al hacer búsquedas de hardware certificado, dando como resultado tiempos de respuestas más rápidos sobre la compatibilidad del hardware durante un proceso de migración.

SICEH brinda un conjunto de servicios web, los cuales otorgan interoperabilidad con otros sistemas como SistDCH para la recopilación de datos permitiendo una actualización constante de la información del hardware.

A la hora de importar hardware a Cuba, utilizar esta herramienta resulta beneficioso, pues a través de la información almacenada facilitaría conocer si el hardware que se desea importar está certificado con determinado sistema operativo, para evitar a largo plazo problemas de incompatibilidad de hardware.

3.5 Conclusiones

En este capítulo se establecen los ciclos iterativos de la implementación mediante la planificación de releases. Con la automatización de las pruebas funcionales se facilita el proceso de desarrollo de una manera ágil y fiable, influyendo en la rápida corrección de errores en las funcionalidades implementadas. Se garantiza la calidad y eficacia de la aplicación realizada, cumpliendo así con el levantamiento de requisitos.

Conclusiones

La presente investigación tiene un papel fundamental en la definición e implementación del Sistema de Certificación y Homologación de Hardware para Nova:

1. Se obtuvieron conocimientos teóricos sobre los distintos sistemas de certificación de hardware para SWL y conceptos fundamentales que permitieron crear la propuesta de solución.
2. Se definieron los requisitos que permitieron definir las capacidades y cualidades que el sistema presenta.
3. Con la realización del diseño de se demostró que la arquitectura definida soporta el desarrollo del sistema.
4. Mediante la implementación del sistema propuesto se obtuvo un sistema que se integra a la PMSWL para la gestión de la información referente a la compatibilidad de hardware que se pretende migrar con el sistema operativo Nova.
5. A partir de pruebas a las funcionalidades implementadas se comprobó que el sistema se encuentra listo para ser usado en procesos de migración reales.

Al finalizar se obtiene una investigación sustentada en elementos teóricos, todos los artefactos definidos dentro del proceso de desarrollo de software seleccionado y un sistema que certifica y homologa hardware para Nova integrado a la plataforma de migración, por lo que se puede decir que se cumplió con el objetivo principal del trabajo, obteniéndose los resultados esperados.

Recomendaciones

Como resultado de la investigación y elementos a tener en cuenta, se hacen las siguientes recomendaciones:

1. Crear dentro de SICEH una aplicación destinada a los usuarios finales implicados en la migración, independiente del sistema de interfaz único de la plataforma de migración.
2. Crear un módulo de estadísticas dentro del sistema para el análisis de compatibilidad de hardware.
3. Crear un módulo de generación de reportes para certificaciones y homologaciones.

Glosario de términos

API Application Programming Interface. Interfaz de programación de aplicaciones con un conjunto de funciones y procedimientos que se ofrece como una biblioteca para ser utilizado por otro software como una capa de abstracción.

CGI Common Gateway Interface. Importante tecnología de la World Wide Web que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa.

Framework Es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.

GITI Gestión de Inventarios de Tecnología Informática . Personalización de OCS Inventory NG: sistema de gestión de inventarios que permiten un control de la información de cualquier ordenador, brindando y almacenando información sobre el hardware y software del mismo.

GPL GNU General Public License . Licencia creada por la Free Software Foundation en 1989, orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

IDEF0 Integration Definition For Funcion Modeling . Metodología para representar de manera estructurada y jerárquica las actividades que conforman un sistema o empresa y los objetos o datos que soportan la interacción de esas

actividades.

MAGM Módulo de Automatización de la Gestión de la Migración . Módulo de la SINPM encargado de gestionar todo el proceso de la migración en este sistema guiado por los procesos descritos en la Metodología Cubana de Migración.

SINPM Sistema de Interfaces del Núcleo de la Plataforma de Migración. Permite la integración y gestión de todos los sistemas de la PMSWL en un sistema centralizado.

SistDCH Sistema de Diagnóstico y Compatibilidad de Hardware . Software que permite extraer información de hardware en una computadora y determinar si es compatible con determinada distribución de GNU/Linux.

Referencias bibliográficas

[1] BISKE, Todd: *SOA Governance*, Packt Publishing, 2008, ISBN 1847195865.

[2] CANONICAL: *Certification Programme Guide*, Inf. téc., Canonical, 2011.
Disponible en: http://www.ubuntu.com/certification/

[3] CESLCAM: Guía de referencia tecnológica para empresas tic, 2010.
Disponible en: http://www.ceslcam.com

[4] ÁVILA CRUZ, Jorge Américo: *Sistema Automatizado para la Selección de Hardware Compatible con Software Libre*, Tesis, Universidad de las Ciencias Informáticas, 2010.

[5] CURTIS, Keith: After the software wars, 2009.
Disponible en: http://keithcu.com/SoftwareWars

[6] FSF: 7 pecados del windows 7, 2009.
Disponible en: http://es.windows7sins.org/privacy/

[7] GARLAN, D. y M. SHAW: An introduction to software architecture, Advances in software engineering and knowledge engineering, 1(1993), págs. 1–40.

[8] GIL, Sandra Victoria Hurtado: *Representación de la arquitectura de software usando UML*, Tesis, Universidad ICESI, 2003.

[9] GNU, Proyecto: Proyecto gnu. la definición de software libre, 2001.
Disponible en: http://www.gnu.org/philosophy/free-sw.es.html

[10] GROUP, The PHP: *PHP for enterprise/business White Paper*, Inf. téc., The PHP Group, 2010.

[11] KIPROP, Kiptoo A.: Certification of software and hardware, Formal Methods for Fun and Profit, (2005), summer Semester 2005.

[12] LÓPEZ, Emilio Suri y Carlos Rafael Galán CABELLO: *Propuesta de Centro de Certificación y Homologación de Hardware para Software Libre*, Tesis, Universidad de las Ciencias Informáticas, 2009.

[13] MONTSERRAT, Culebro Juárez, Gómez Herrera Wendy GUADALUPE y Torres Sánchez SUSANA: *Comparación entre el software libre y el software propietario. Sus ventajas y sus desventajas*, Tesis, Universidad Nacional Autónoma de México, may 2006.

[14] NIETO, Jesús Javier Estepa: *Software Libre para el Desarrollo del Tercer Mundo*, Tesis, Universidad de Granada, Julio 2007.

[15] ORTIZ, Bruno: *En solo 40 años Internet ha modificado nuestro mundo*, 2009.

[16] PRESSMAN, Roger S.: *Ingeniería del Software. Un enfoque práctico*, Mc Graw Hill, 5 edón., 2001.

[17] RAE: Diccionario de la lengua española, 2011, vigésima Segunda edición.
Disponible en: http://buscon.rae.es/draeI/

[18] REYNOSO, Carlos Billy: Introducción a la arquitectura de software, 04 2004.
Disponible en: http://carlosreynoso.com.ar

[19] RICHARDS, Robert: *Pro PHP XML and Web Services*, Apress, 2006, ISBN 978-1-59059-633-3.

[20] RIGGS, Simon y Hannu KROSING: *PostgreSQL 9 Administration Cookbook*, PACKT, oct 2010, ISBN 978-1-849510-28-8.

[21] ROMERO, Gladys Marsi Peñalver: *MA-GMPR-UR2. Metodología ágil para proyectos de software libre.*, Tesis, Universidad de las Ciencias Informáticas, 10 2008, versión 0.2.

[22] ROMERO, Gladys Marsi Peñalver y Abel Meneses ABAD: *SXP, metodología ágil para proyectos de software libre*, Tesis, Universidad de las Ciencias Informáticas, 2009.

[23] SERVON, L.: *Bridging the Digital Divide. Technology, community and public policy*, Blackwell Publishing, 2002.

[24] STALLMAN, Richard M.: *Software libre para una sociedad libre*, GNU Press, 2004, ISBN 84-933555-1-8.

[25] DE LA TEJERA, Inglis Pavón y Marisol Lopez VELAZQUEZ: Ventajas y desventajas de la migración de los sistemas propietarios a los sistemas gnu/open-source para cuba., en *IV Taller Internacional de Software Libre y estándares abiertos de software.*, 2007, 7.

[26] VILLAZÓN, Yoandy Pérez: *Metodología para la Migración a Software Libre de las Universidades del Ministerio de Educación Superior*, Tesis, Universidad de las Ciencias Informáticas, 2008.

[27] VITIER, Abel García y Jailen García GONZÁLEZ: *Integración de los sistemas de la Plataforma de Migración a Software Libre y Código Abierto*, Tesis, Universidad de las Ciencias Informáticas, 2011.

[28] W3C: Definición de servicios web, 2011.
Disponible en: http://www.w3.org/

[29] ZANINOTTO, F. y F. POTENCIER: *A Gentle Introduction to symfony 1.4*, Sensio SA, 2011.

Bibliografía consultada

[1] BOWMAN, Judith S., et al.: *The Practical SQL Handbook*, Addison-Wesley, 3 edón., oct 1996, ISBN 0201447878.

[2] BRADENBAUGH, Jerry: *Programación de aplicaciones Web con JavaScript*, Aplicaciones Javascript, O'Reilly, feb 2000, ISBN 84-41 5-1070-9.

[3] CAMPO, J.C.A., R.A.T. BENÍTEZ, F. ALEXANDER, R. ORDÓÑEZ, J.R. RÍOS, J.A.B. MEDINA y J.J.F. RODRÍGUEZ: *Gobierno electrónico: Acortando la brecha digital*, Universidad Santiago de Cali, 2009.

[4] CASTLEDINE, Earle y Craig SHARKIE: *jQuery: Novice to Ninja*, SitePoint Pty. Ltd., 2010, ISBN 978-0-9805768-5-6.

[5] CHAFFER, Jonathan y KARL SWEDBERG: *jQuery Reference Guide*, Pack, ago 2007, ISBN 978-1-847193-81-0.

[6] DOUGLAS, Korry y Susan DOUGLAS: *PostgreSQL: The comprehensive guide to building, programming, and administering PostgreSQL databases*, Sams Publishing, 2 edón., jul 2005, ISBN 0-672-32756-2.

[7] GRANADO, Luis Miguel Cabezas: *Manual Imprescindible de PHP5*, Anaya, 2004, ISBN 84-415-1785-1.

[8] HEOW EIDE-GOODMAN, Steven D. Nowicki y Alec COVE: *Professional PHP5*, Wiley Publishing, Inc., 2005, ISBN 0-7645-7282-2.

[9] HERNÁNDEZ, Lisandra Cala: *Propuesta de Integración y Nuevas Herramientas para el Desarrollo de La Plataforma Cubana de Migración a Software Libre*, Tesis, Universidad de las Ciencias Informáticas, 2010.

[10] JACOBSON, L.: *Object-Oriented Software Engineering; A Use Case Driven Aproach*, Adison-Wesley, 1992.

[11] JONATHAN H. WAGE, Roman Borschel y Guilherme BLANCO: *Doctrine ORM for PHP 1.2*, 1.2 edón., feb 2010, ISBN 2918390038.

[12] LABS, Sensio: Más con symfony, Mar 2010.
Disponible en: http://librosweb.es/mas-con-symfony/

[13] LEWIS, G: *What is Software Engineering?*, DataPro, feb 1994.

[14] MEHDI ACHOUR, Friedhelm Betz et al: *Manual de PHP*, PHP Documentation Group, sep 2011.

[15] MOMJIAN, Bruce: *PostgreSQL: Introduction and Concepts*, Addison–Wesley, 2001.

[16] PATERNINA PALACIO, K.: Ingeniería del software, un enfoque práctico, INGENIATOR, 1(2011), nº 1.

[17] POTENCIER, Fabien y François ZANINOTTO: Symfony la guía definitiva, dic 2008.
Disponible en: http://www.librosweb.es/symfony_1_2

[18] POWELL, Gavin: *Beginning Database Design*, Programmer to Programmer, Wiley Publishing, Inc., 2006, ISBN 978-0-7645-7490-0.

[19] PRESSMAN, Roger S.: *Ingeniería de Software. Un enfoque práctico*, Mc Graw Hill, 2006, ISBN 970-10-5473-3.

[20] PÉREZ, Javier Eguíluz: Css avanzado, ene 2009.
Disponible en: http://www.librosweb.es/css_avanzado

[21] — Introducción a javascript, feb 2008.
Disponible en: http://www.librosweb.es/javascript

[22] TIDWELL, Jenifer: *Designing Interfaces*, O'Reilly, Nov 2005, ISBN 0-596-00803-1.

[23] ULLMAN, Larry: *VISUAL QUICKPRO GUIDE: PHP 5 ADVANCED*, Peachpit Press, 2007, ISBN 0-321-37601-3.

[24] VILALTA, Josep: *UML Guía Visual*, Vilalta Consultores, sep 2001.

[25] VILLAZÓN, Yoandy Pérez, Ramón Paumier SAMÓN y Abel Meneses ABAD: *Guía Cubana para la Migración a Software Libre*, Universidad de las Ciencias Informáticas, 2009.

A

Anexos

Servicios Web

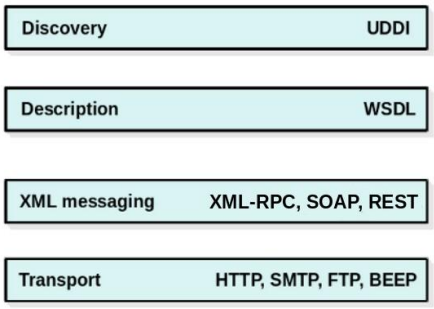

Figura A.1: Pila de protocolos de Servicios Web

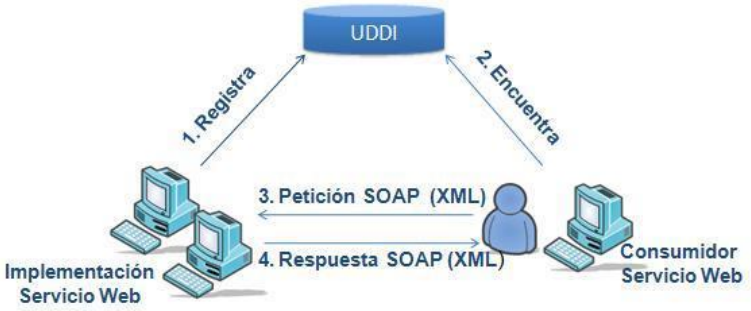

Figura A.2: Funcionamiento arquitectura SOA

Modelo de negocio

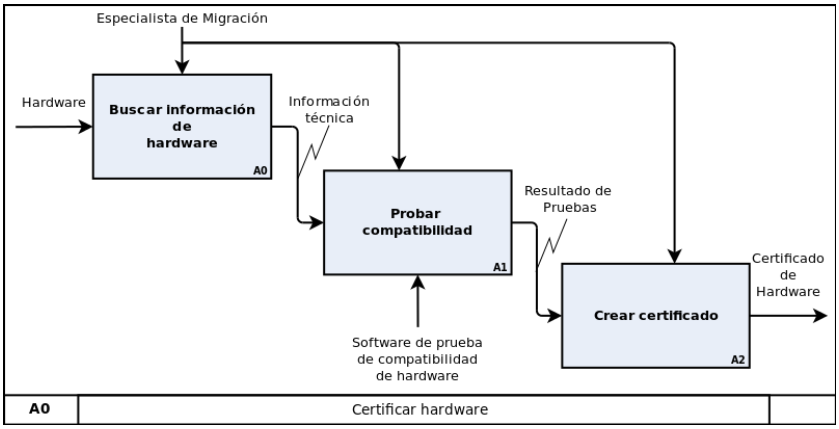

Figura A.3: Proceso de certificación detallado

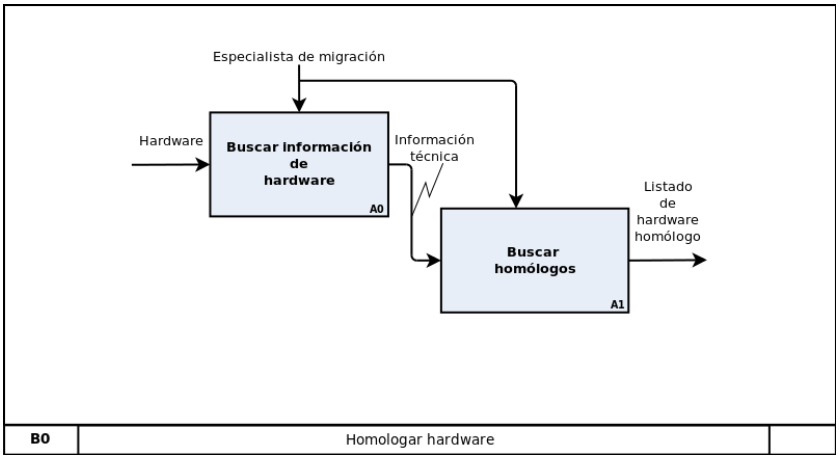

Figura A.4: Proceso de homologación detallado

Prototipos de interface

Figura A.5: Interfaz de Historia de Usuario: Gestionar fabricante

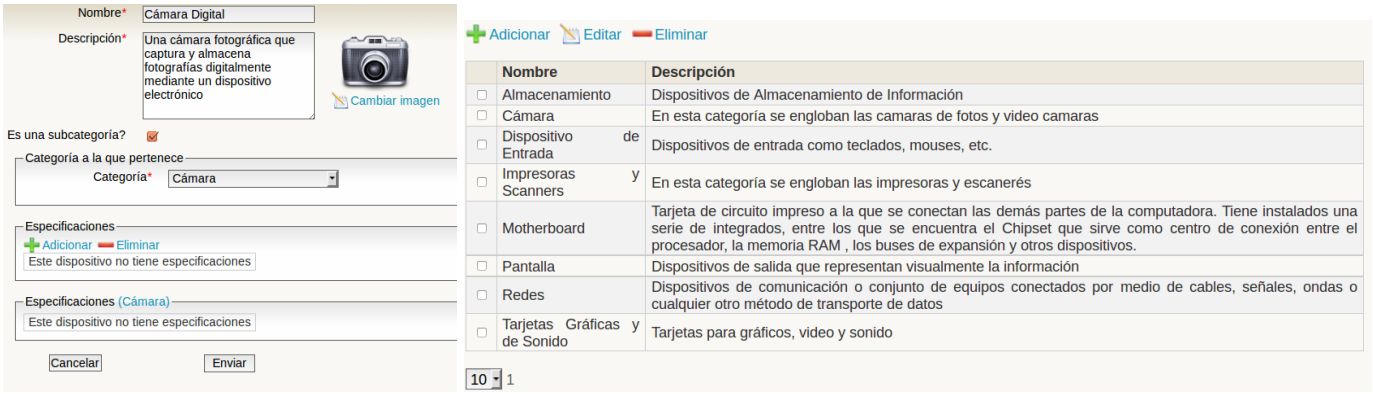

Figura A.6: Interfaz de Historia de Usuario: Gestionar categorías de hardware

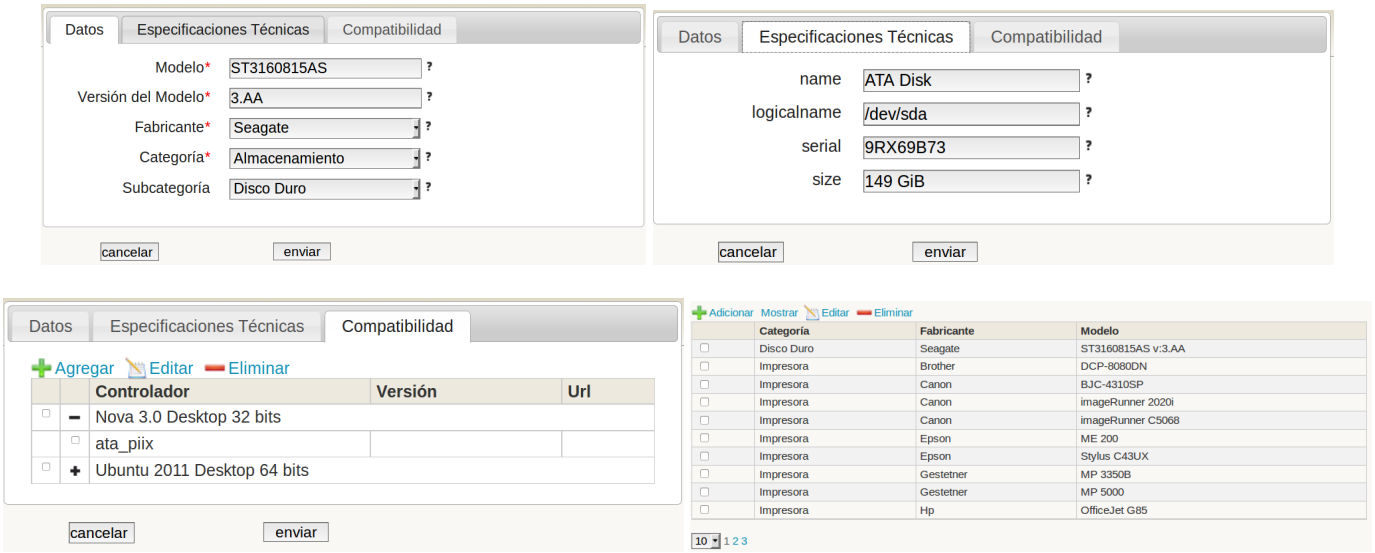

Figura A.7: Interfaz de Historia de Usuario: Gestionar hardware

Figura A.8: Interfaz de Historia de Usuario: Gestionar hardware. Mostrar hardware

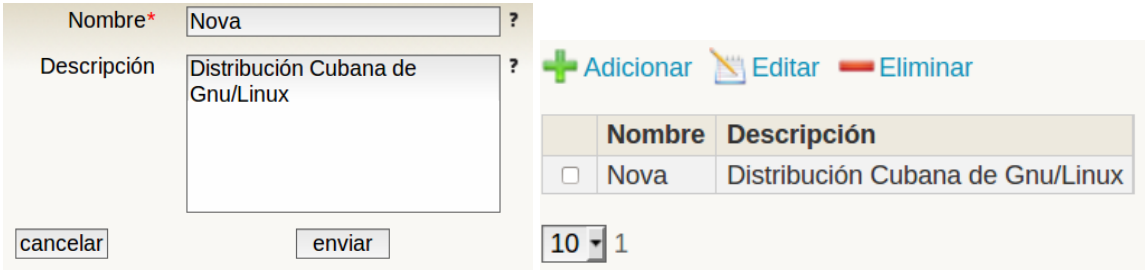

Figura A.9: Interfaz de Historia de Usuario: Gestionar distribución de sistema operativo

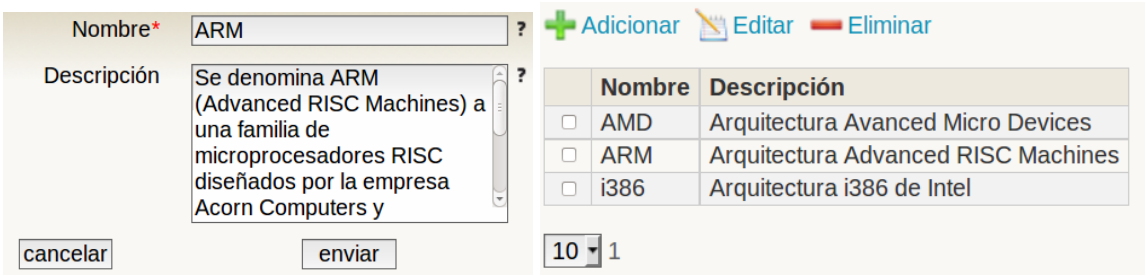

Figura A.10: Interfaz de Historia de Usuario: Gestionar arquitectura de sistema operativo

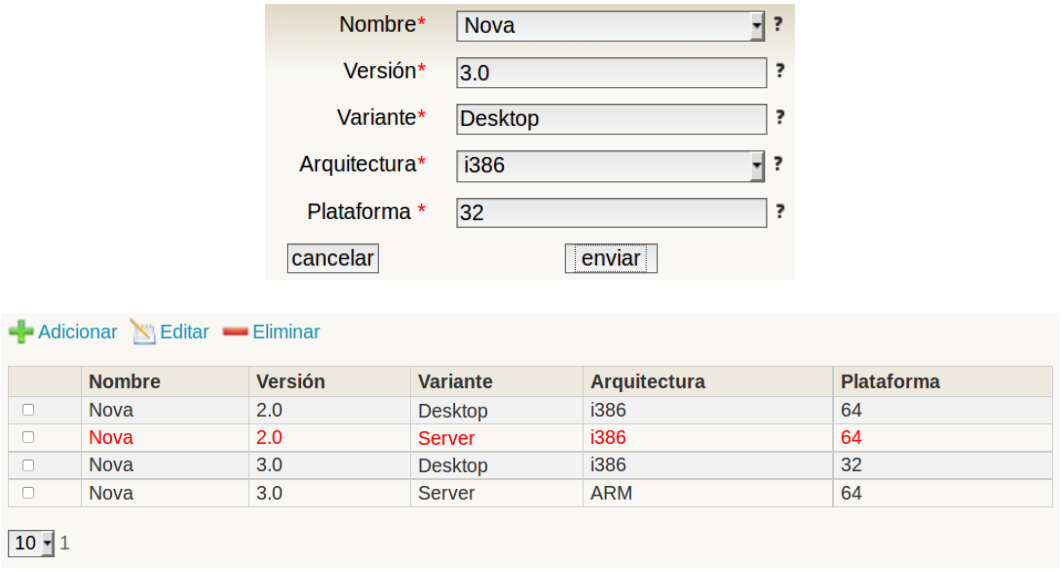

Figura A.11: Interfaz de Historia de Usuario: Gestionar sistema operativo

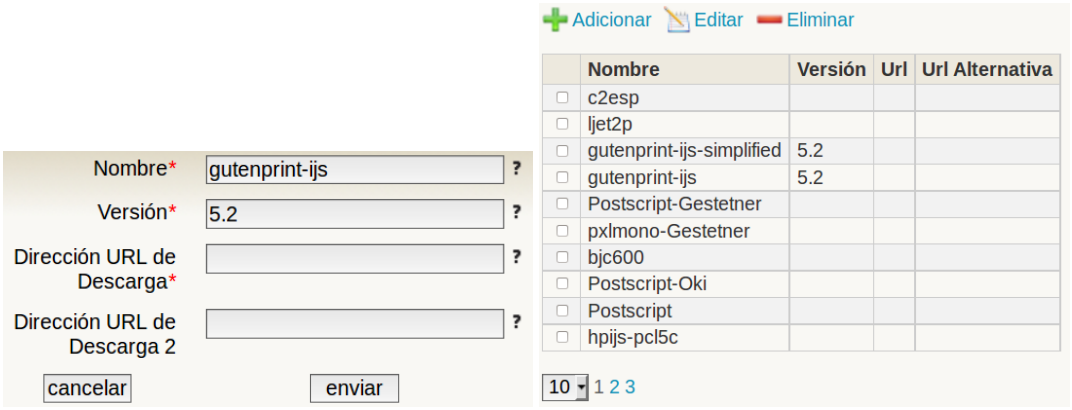

Figura A.12: Interfaz de Historia de Usuario: Gestionar driver

Figura A.13: Interfaz de Historia de Usuario: Gestionar categorías de certificados de hardware

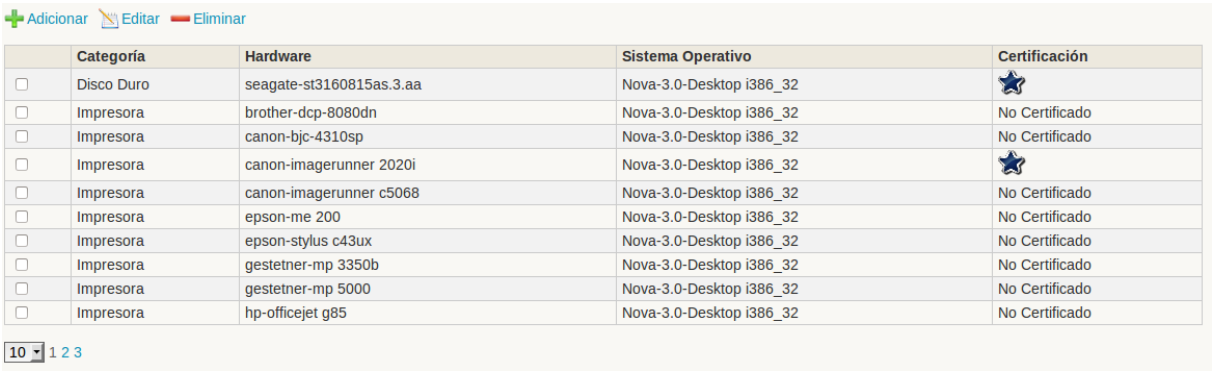

Figura A.14: Interfaz de Historia de Usuario: Gestionar certificados de hardware

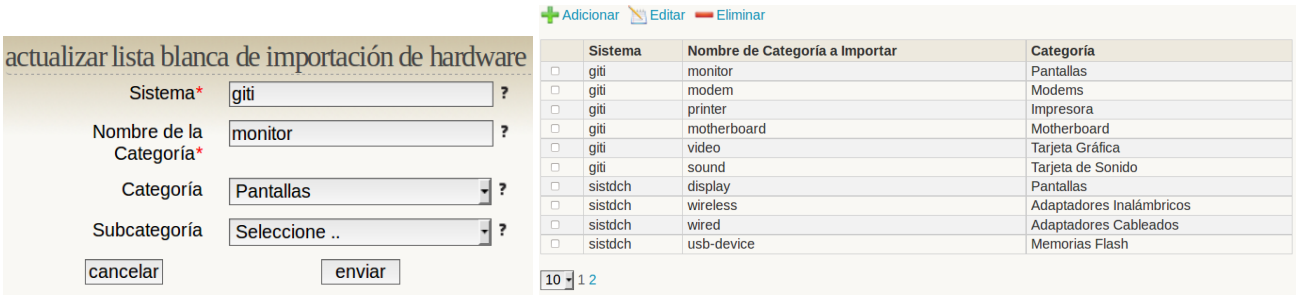

Figura A.15: Interfaz de Historia de Usuario: Gestionar reglas de importación de hardware

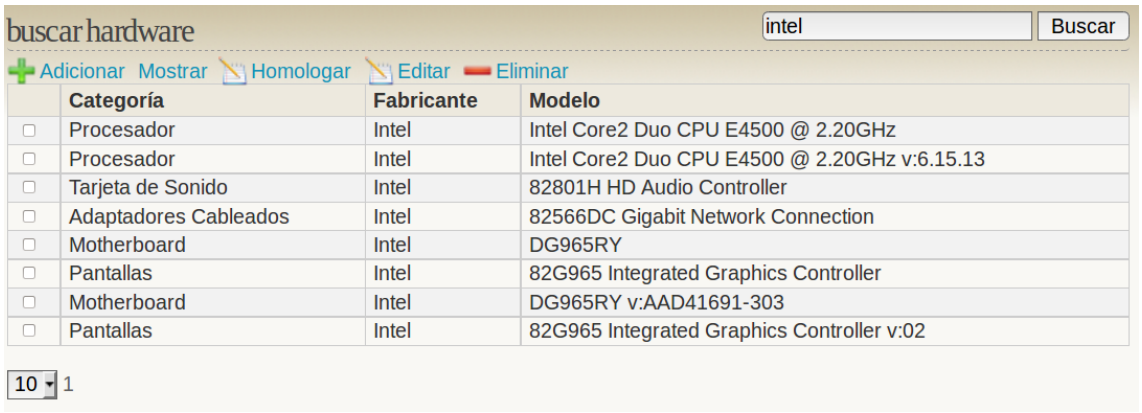

Figura A.16: Interfaz de Historia de Usuario: Buscar hardware

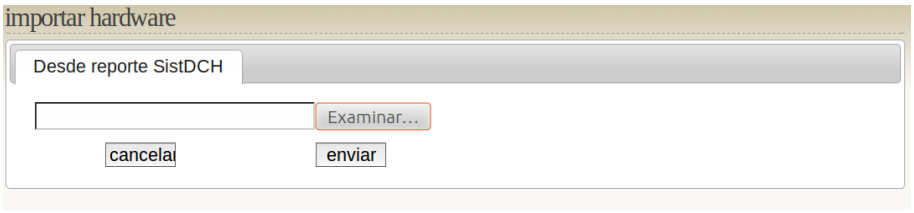

Figura A.17: Interfaz de Historia de Usuario: Importar reporte de hardware en XML

Figura A.18: Interfaz de Historia de Usuario: Buscar hardware homólogo

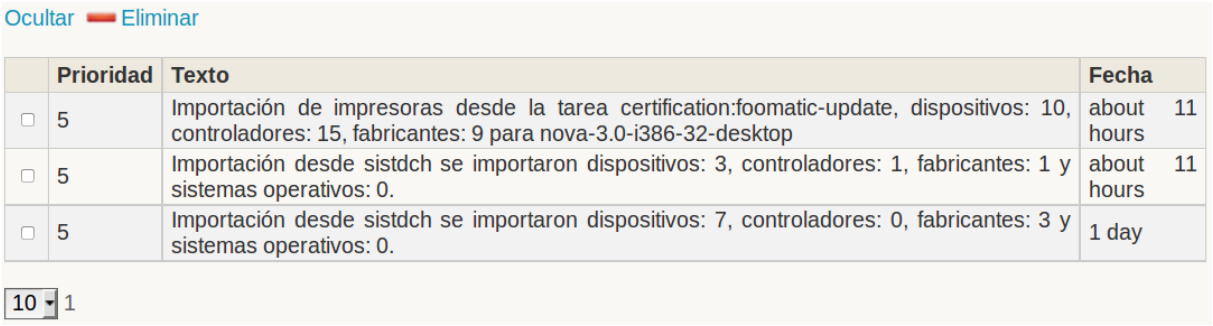

Figura A.19: Interfaz de Historia de Usuario: Gestionar notificaciones

Figura A.20: Interfaz de Historia de Usuario: Gestionar comentarios

Figura A.21: Interfaz de Historia de Usuario: Evaluar compatibilidad de hardware

Figura A.22: Interfaz de Historia de Usuario: Descargar SistDCH

Diagramas de ingeniería

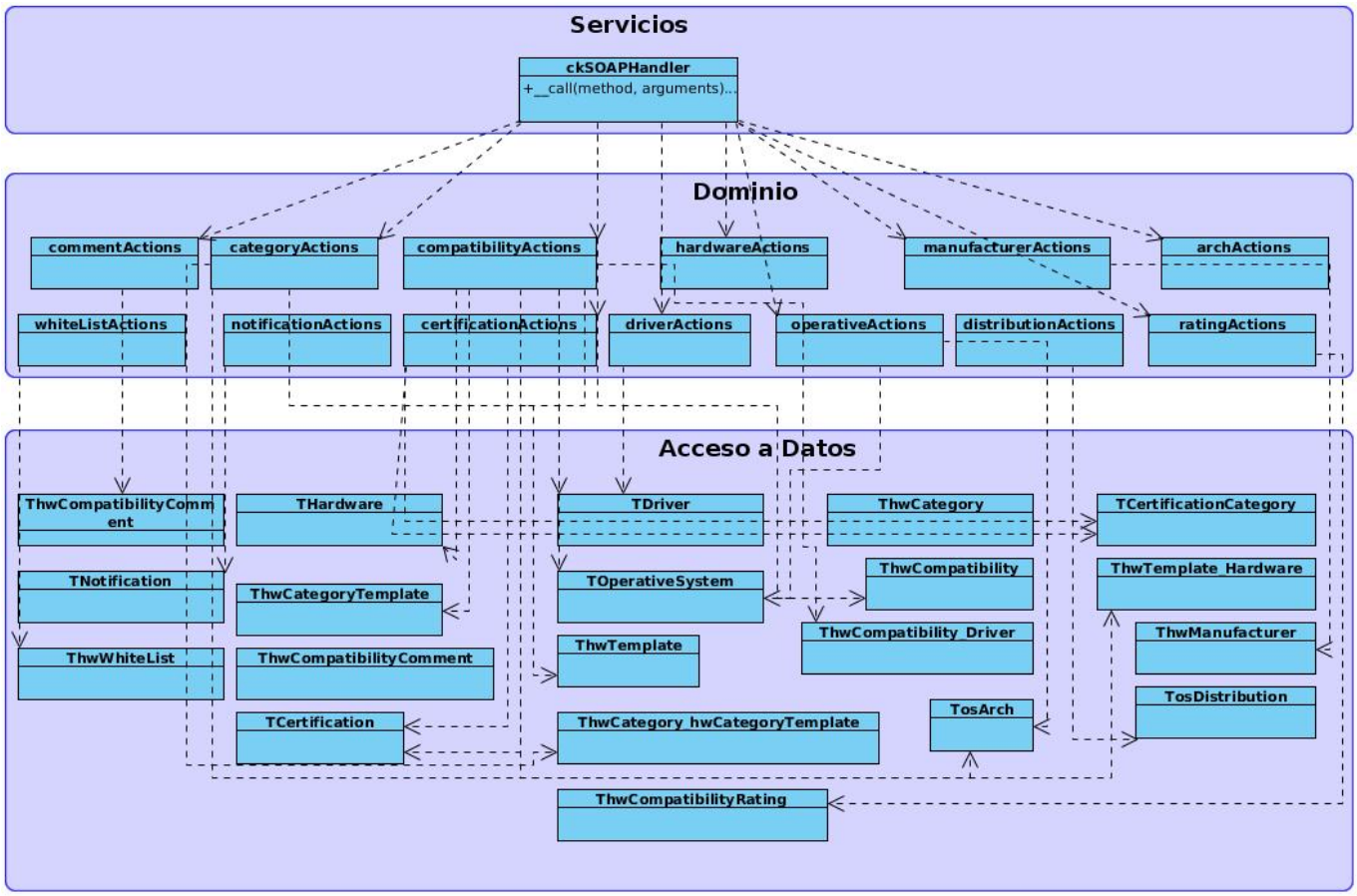

Figura A.23: Diagrama de Clase del Diseño de SICEH

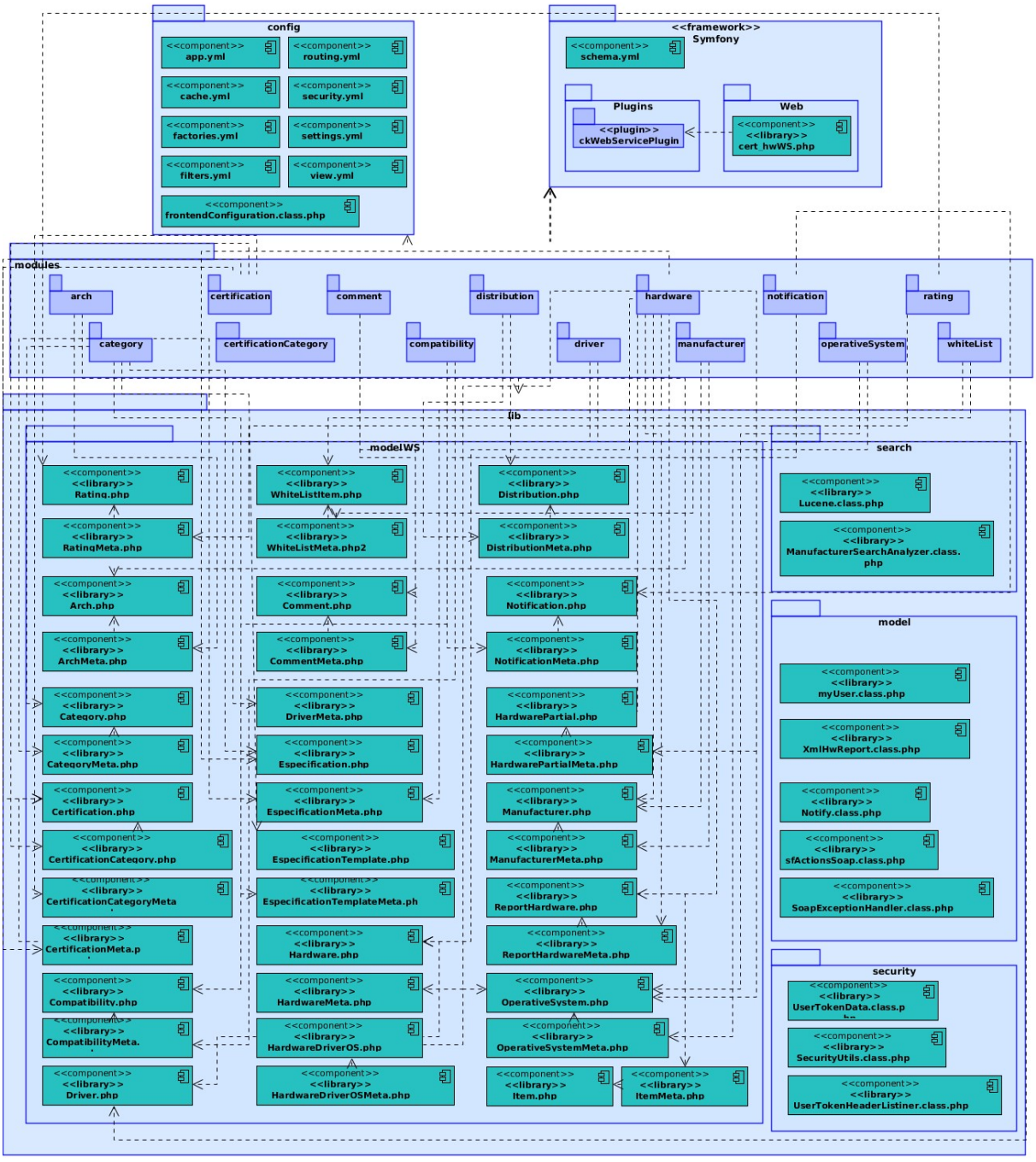

Figura A.24: Diagrama de Componentes de SICEH

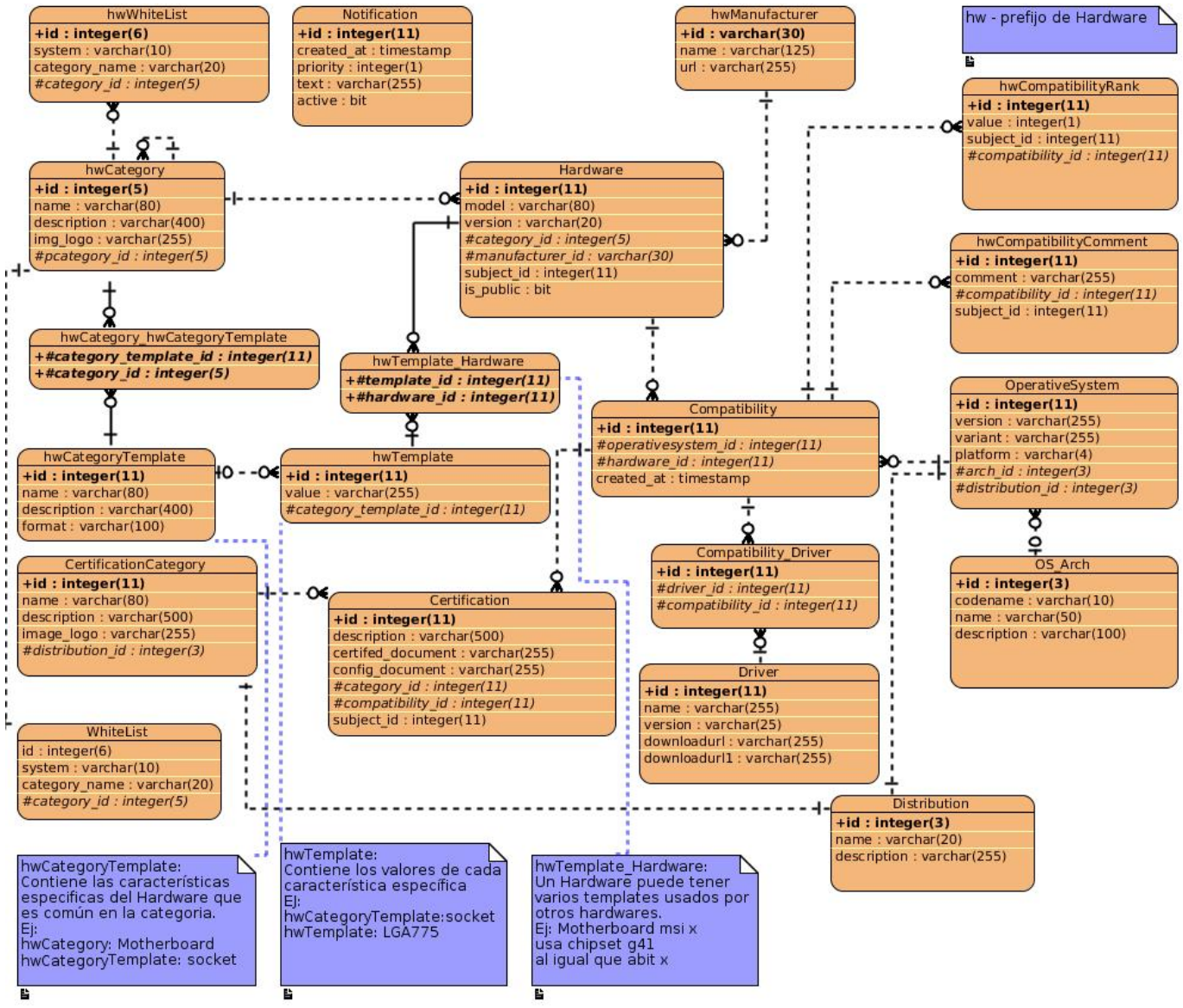

Figura A.25: Diagrama Entidad - Relación de SICEH

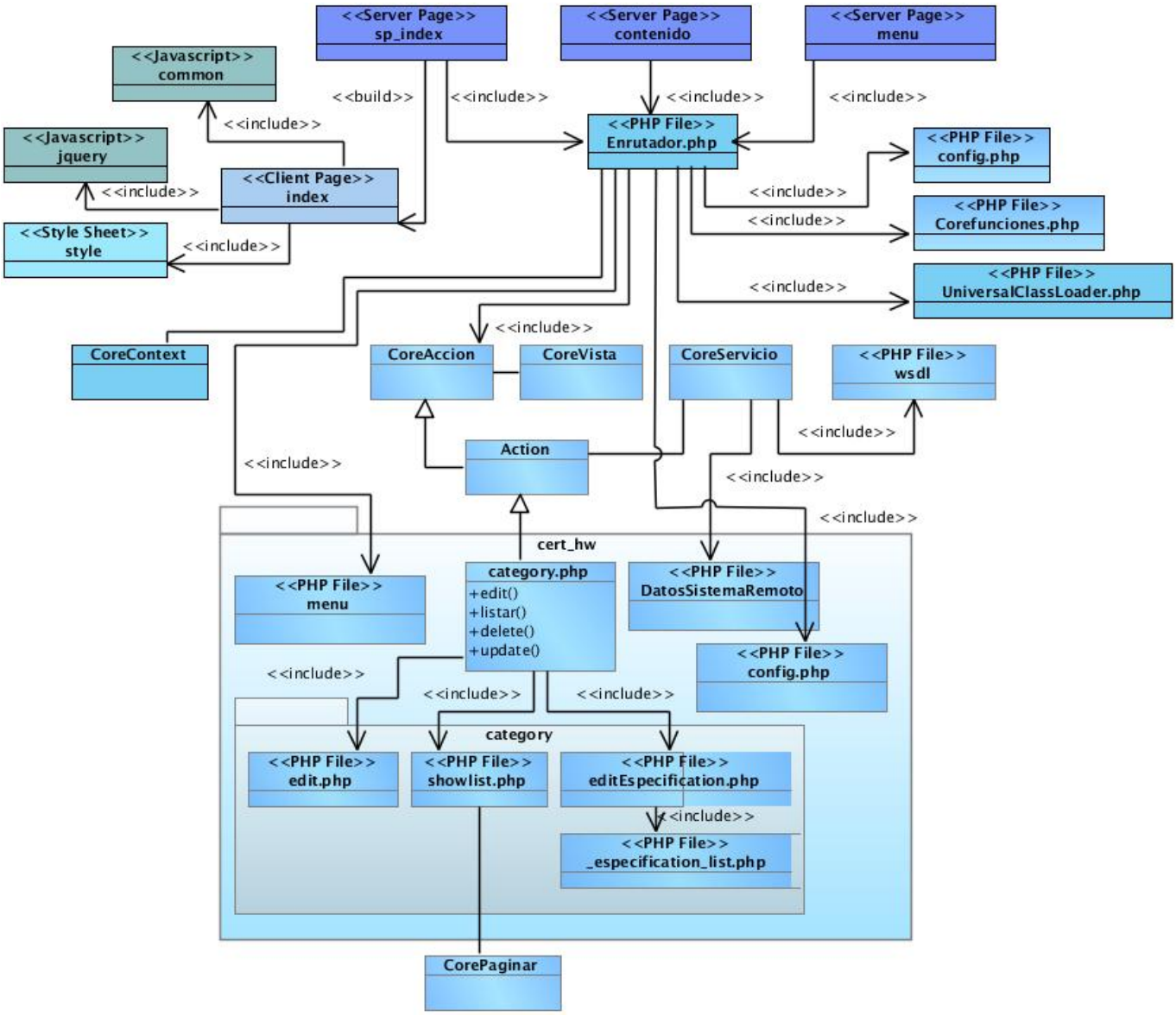

Figura A.26: Diagrama de Clase: Módulo de interfaz de usuario de SICEH. HU Gestionar Categorías

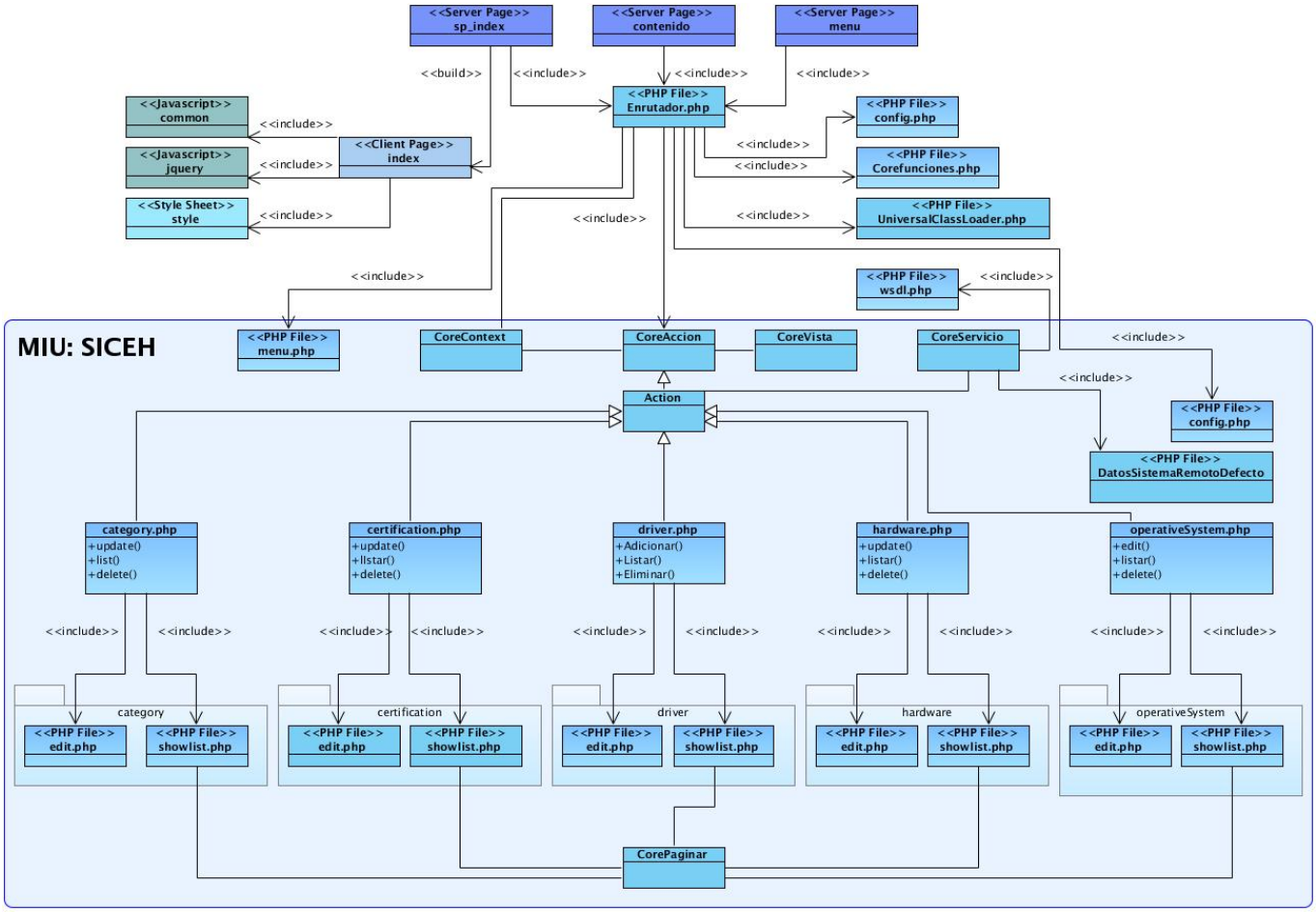

Figura A.27: Diagrama de Clase: Módulo de interfaz de usuario de SICEH

Symfony

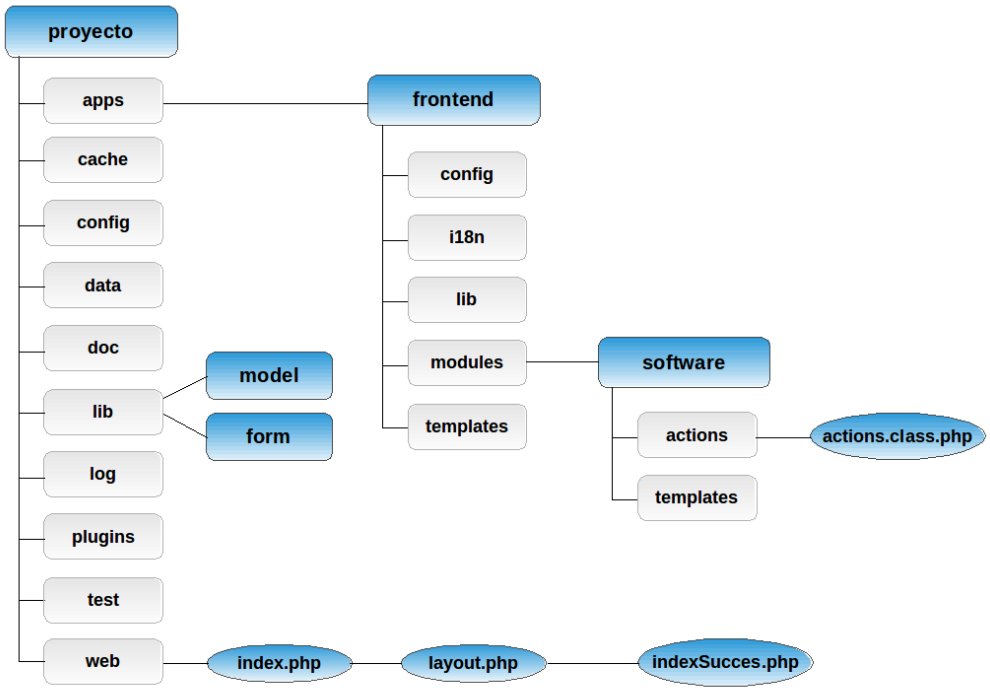

Figura A.28: Estructura de directorios de un proyecto Symfony 1.4.x

```yaml
all:
  enable_soap_parameter: off
  # tiempo que dura una sesión
  sessionexpirationtime: 20
  homologation:
    result_limit: 10
  search:
    lucene:
      enable:   true
      dir:      %SF_DATA_DIR%/lucene
      min_prefix_length: 2
  images:
    hardware:
      category: %SF_WEB_DIR%/images/hardware/category
    certification:
      category: %SF_WEB_DIR%/images/certification/category
    options:
      all:
        max_size: 2MB
        type:     [png,jpg]
      logo:
        # 1MB
        max_size: 1048576
        max_height: 200
        max_width:  200
        crop:  true
        type: [png]
  documents:
    root:          %SF_WEB_DIR%
    certification: /uploads/assets
    options:
      max_size: 3MB
      type:     [doc,odt,pdf]
  security:
    # sistema de seguridad
    ssnpm:
      # dominio de la aplicación
      domain:   http://127.0.0.1/ssnpm/
      # url del wsdl de la aplicación
      wsdl:     http://127.0.0.1/ssnpm/ssnpmWS.wsdl
      # usuario del sistema y contraseña
      user:     cert_hw
      passwd:   cert_hw

```

```yaml
soap:
  enable_soap_parameter: on
  ck_web_service_plugin:
    wsdl: %SF_WEB_DIR%/cert_hwWS.wsdl
    handler: cert_hwWSHandler
    soap_headers:
      Security:
        class: SecurityData
      UserToken:
        class: UserTokenData
    soap_options:
      encoding:     utf-8
      soap_version: <?php echoln(SOAP_1_2) ?>
      persist: <?php echoln(SOAP_PERSISTENCE_SESSION); ?>
      render: off
      # Mapeo de Objetos Soap a clases
      classmap:
        # Arreglos de datos nativos
        StringArray:      ckGenericArray
        DoubleArray:      ckGenericArray
        IntArray:         ckGenericArray
        # Modelo de Objetos
        Category:             Category
        CategoryArray:        ckGenericArray
        CategoryMeta:         CategoryMeta
        Certification:        Certification
        CertificationArray:   ckGenericArray
        CertificationMeta:    CertificationMeta
        CertificationCategory:        CertificationCategory
        CertificationCategoryArray:   ckGenericArray
        CertificationCategoryMeta:    CertificationCategoryMeta
        Compatibility:        Compatibility
        CompatibilityArray:   ckGenericArray
        CompatibilityMeta:        CompatibilityMeta
        CompatibilityOs:      CompatibilityOs
        CompatibilityOsArray:   ckGenericArray
        Distribution:         Distribution
        DistributionArray:    ckGenericArray
        DistributionMeta:     DistributionMeta
        # ... demás modelos de objetos
```

Figura A.29: Configuración de Aplicación SICEH, fichero app.yml de Symfony

Pruebas funcionales

Código A.1: Código de test funcional mediante Lime

```
<?php
include(dirname(__FILE__) . '/../../bootstrap/functionalSoapTest.php');
# opciones soap para mapeo de clases
$options = sfConfig::get('app_ck_web_service_plugin_soap_options');
$options['classmap']['UserTokenData'] = '\\UserTokenData';

$soapClient = new ckTestSoapClient($options); #cliente soap de pruebas.
$t = $soapClient->test(1); #plan de pruebas.

$t->info('Agregar objeto category');

$category = new Category(null, null, gen_str(), gen_str(0, 0, true));

# token de seguridad mediante cabeceras Soap, para pruebas.
$soapClient->addRequestHeader('UserToken', new \UserTokenData('admin@token'));
$soapClient
          # llamada Soap.
        ->addCategory($category)
          # comprueba que no existan excepciones Soap.
        ->isFaultEmpty()
          # comprueba que el resultado sea un integer.
        ->isType('', 'integer')
;
```

```
1..10
> Adicionar hardware
# addHardware(...)
ok 1 - response object . is a integer
> Obtener un hardware
# getHardware(...)
ok 2 - response object list.0.id is 411
ok 3 - response object list.0.model is gmj
> Actualizar Hardware
# updateHardware(...)
ok 4 - response object . is a boolean
# getHardware(...)
ok 5 - response object list.0.id is 411
ok 6 - response object list.0.model is MMX-GT 500
> Listar y Filtrar hardware
# listHardwarePartial(...)
ok 7 - response object list.0.id is 411
ok 8 - response object list.0.model is MMX-GT 500
> Eliminar una lista de hardware
# deleteHardware(...)
ok 9 - response object . is a boolean
ok 10 - response object . is 1
# Looks like everything went fine.
```

Figura A.30: Caso de Prueba: HU Gestionar hardware

```
1..9
> Agregar Sistema Operativo
# addOperativeSystem(...)
ok 1 - response object . is a integer
> Agregar Sistema Operativo existente
# addOperativeSystem(...)
ok 2 - response soap fault message is element already exists
> Obtener un Sistema Operativo
# getOperativeSystem(...)
ok 3 - response object . is a OperativeSystemMeta
ok 4 - response object list.0.id is 71
ok 5 - response object length is 1
> Actualizar una Sistema Operativo
# updateOperativeSystem(...)
ok 6 - response object . is a boolean
ok 7 - response object . is 1
> Eliminar una Sistema Operativo
# deleteOperativeSystem(...)
ok 8 - response object . is a boolean
ok 9 - response object . is 1
# Looks like everything went fine.
```

Figura A.31: Caso de Prueba: HU Gestionar sistema operativo

```
1..9
> Agregar una Categoría de Certificación
# addCertificationCategory(...)
ok 1 - response object . is a integer
> Agregar una Certificación
# addCertification(...)
ok 2 - response object . is a integer
> Obtener una Certificación
# getCertification(...)
ok 3 - response object . is a Certification
ok 4 - response object category is 27
ok 5 - ->getCertification() documento creado
> Actualizar una Certificación
# updateCertification(...)
ok 6 - response object . is a boolean
ok 7 - response object . is 1
> Eliminar una Certificación
# deleteCertification(...)
ok 8 - response object . is a boolean
ok 9 - response object . is 1
# Looks like everything went fine.
```

Figura A.32: Caso de Prueba: HU Gestionar certificación

```
1..4
> Importar hardware
# importHardware(...)
ok 1 - response object . is a ckGenericArray
ok 2 - response object . contains 40 elements
# getHardware(...)
ok 3 - response object list.0 is a Hardware
ok 4 - response object list.0.id is 471
# Looks like everything went fine.
```

Figura A.33: Caso de Prueba: HU Importar hardware

```
1..5
> Homologar Hardware
# homologateHardware(...)
ok 1 - response object . is a HardwarePartialMeta
ok 2 - response object list is a ckGenericArray
ok 3 - response object list.0.id is 544
ok 4 - response object list.1.id is 543
ok 5 - response object list.2.id is 545
# Looks like everything went fine.
```

Figura A.34: Caso de Prueba: HU Buscar hardware homólogo

Estructura de principales objetos SOAP

Código A.2: Clase HardwareReportMeta

```php
<?php
class HardwareReportMeta {
  /** @var string $system */
  public $system;
  /** @var HardwareReport[] $list */
  public $list;
  /** @var int $length */
  public $length;
}
?>
```

Código A.3: Clase HardwareReport

```php
<?php
class HardwareReport{
  /** @var string $operativesystem */
  public $operativesystem;
  /** @var string $category */
  public $category;
  /** @var string $manufacturer */
  public $manufacturer;
  /** @var string $model */
  public $model;
  /** @var string $version */
  public $version;
  /** @var Item[] $especifications */
  public $especifications;
  /** @var Driver[] $drivers */
  public $drivers;
```

```
}
?>
```

Código A.4: Clase Driver

```
<?php
class Driver {
  /** @var int $id */
  public $id;
/** @var string $name */
  public $name;
  /** @var string $version */
  public $version;
  /** @var string $downloadurl */
  public $downloadurl;
  /** @var string $downloadurl2 */
  public $downloadurl2;
}
?>
```

Código A.5: Clase Item

```
<?php
class Item {
  /** @var string $name */
  public $name;
  /** @var string $value */
  public $value;
}
?>
```