Universidad de las Ciencias Informáticas
Facultad 1

Título: *Aplicación web del DGM para la administración centralizada de dispositivos de hardware en ambientes multiplataforma.*

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

Autor(es): *Lien Morell Baró*

Yasiel Lora Rodríguez

Tutor(es): *Msc. Adrian Alberto Machado Cento*

Ing. Iván Campos Cesar

La Habana, junio del 2015

Declaración de autoría

Declaramos que somos los únicos autores del trabajo titulado:

Aplicación web del DGM para la administración centralizada de dispositivos de hardware en ambientes multiplataforma a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____.

Autor: Lien Morell Baró

_____

Autor: Yasiel Lora Rodríguez

_____

Tutor: Msc. Adrian Machado Cento

_____

Tutor: Ing. Iván Campos Cesar

_____

Resumen

Controlar y administrar dispositivos del hardware siempre ha sido la necesidad de las grandes instituciones que lo utilizan. Para ello se crearon sistemas que permiten la administración centralizada de dispositivos, que brindan robustez, comodidad y ayudan a preservar la seguridad de los mismos.

En el Centro de Identificación y Seguridad Digital (CISED) de conjunto con el Ministerio del Interior (MININT) se llevó a cabo el Device Grid Manager (DGM) aplicación que permite el control e interacción centralizado de dispositivos de hardware en aplicaciones web. Esta aplicación permite al Sistema Único de Identificación Nacional de la República de Cuba (SUIN) la comunicación e interacción con los dispositivos de hardware instalados en las estaciones de trabajo y a partir de ellos obtener los datos necesarios para llevar a cabo los procesos de captación biométrica. En la actualidad este se encuentra desarrolla con tecnologías privativas impidiendo que pueda ser utilizado en otro ambiente de trabajo.

Con el fin de resolver los problemas existentes que presenta este sistema, se decidió desarrollar una aplicación web multiplataforma que sea capaz de administrar de forma centralizada los dispositivos del hardware. Para ello se lleva a cabo una valoración crítica de los sistemas existentes que permiten la comunicación con dispositivos del hardware, se seleccionan las herramientas necesarias para el desarrollo del sistema y además se diseña, implementa y se válida para comprobar su correcto funcionamiento por medio de pruebas de funcionalidad.

Palabras claves: dispositivos, administración centralizada, hardware, software libre.

Introducción

Con el paso de los años, gran parte de la sociedad a nivel mundial se ha encontrado informatizada y cada minuto que pasa el hombre se hace más dependiente de los ordenadores. Diariamente se realizan nuevos sistemas, con los que son automatizados procesos que influyen en el aumento de la productividad y la eficiencia del trabajo como variable del desarrollo social. Existen empresas e instituciones que poseen y utilizan numerosos recursos de hardware y software, por lo que se hace necesario tener un control de los mismos a través de aplicaciones que permitan gestionarlos, conocer de estadísticas y características de los mismos.

En el Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) de conjunto con el Ministerio del Interior (MININT) se desarrolló un software denominado Device Grid Manager (DGM) que tiene como objetivo llevar a cabo la administración centralizada de dispositivos de hardware y su interacción con aplicaciones web. Este sistema está compuesto por tres subsistemas fundamentales: el Servicio DGM para el manejo local de dispositivos, el Framework JavaScript para la interacción con servicios publicados en la estación de trabajo y el Sistema para el control centralizado de dispositivos.

De ellos, el Sistema para el control centralizado de dispositivos brinda una interfaz visual en forma de aplicación web que permite al usuario gestionar cada uno de los dispositivos de la red así como ver en tiempo real el estado de estos. Permite además gestionar las localizaciones de los dispositivos, obtener reportes de uso y comportamiento de estos; el mismo se caracteriza por el fácil uso, permitiéndose usar en prácticamente cualquier escenario. (1)

Sin embargo esta aplicación web está desarrollada en su mayoría con herramientas privativas y solo puede ser desplegada con servidores Windows que empleen *Internet Information Services* (IIS) lo que provoca incurrir en gastos para licencias de software y esto impide que pueda ser desplegado en cualquier otro ambiente de trabajo. El uso de esta solución ha demostrado que la obtención de reportes por parte del usuario es incompleto, de ahí la necesidad de incluir otros reportes de los dispositivos que se encuentran registrados en el sistema; y a pesar que posee varias funcionalidades, no permite conocer en tiempo real si un ordenador conectado a la red se encuentra encendido o no, y con ello tener un mayor control de los dispositivos que se encuentran dentro del sistema.

Atendiendo la situación planteada y enfocándose en su solución, se propone como **problema de investigación**: ¿Cómo garantizar la administración centralizada de dispositivos de hardware en ambientes multiplataforma?

Definiendo como **objeto de estudio:** La administración de dispositivos de hardware.

Para dar solución al problema se propone como **objetivo general**: Desarrollar una aplicación web para la administración centralizada de dispositivos de hardware en ambientes multiplataforma. Desglosándose este en los siguientes **objetivos específicos**:

- Desarrollar el marco teórico de la investigación para un mayor entendimiento del área de conocimiento.
- Definir las herramientas a utilizar para el desarrollo de la aplicación que se propone.
- Realizar el análisis y diseño de la solución del sistema de componentes para guiar el proceso de implementación.
- Implementar el sistema que permita la administración centralizada de los dispositivos de hardware.
- Realizar pruebas de software al sistema desarrollado para validar su correcto funcionamiento.

Para darle cumplimiento a los objetivos específicos se realizarán las siguientes **tareas de la investigación**:

- Análisis de los principales conceptos asociados a los procesos administración de dispositivos desde aplicaciones web para obtener una base teórica necesaria para el desarrollo de la solución.
- Revisión bibliográfica acerca de soluciones existentes que permiten la administración centralizada de dispositivos de hardware, para determinar fortalezas e insuficiencias en las mismas y valorar la vialidad de su uso.
- Realización de un estudio que permita definir cuáles son las herramientas informáticas y metodologías a usar para el desarrollo de la aplicación.
- Especificación de los requisitos del sistema de componentes para la administración centralizada de dispositivos, con el objetivo de determinar las funcionalidades a implementar.

- Diseño de la solución del sistema de componentes para la administración y el control centralizado.
- Desarrollo de una aplicación web que permita la administración centralizada de los dispositivos de hardware.
- Desarrollo de un módulo que permita la activación o registro de las estaciones de trabajo en el sistema.
- Desarrollo de un componente que garantice mecanismos de autenticación – autorización entre los servicios de forma segura.
- Aplicación de las pruebas de software a la aplicación web para la administración centralizada de dispositivos de hardware.

Para asistir las tareas de la investigación se emplearán los siguientes **métodos científicos:**

Métodos teóricos:

1. **Histórico – lógico**: Permitió realizar un estudio del estado del arte de la problemática; así como analizar las ventajas y desventajas de las herramientas utilizadas actualmente para la administración de dispositivos en aplicaciones web.
2. **Analítico – sintético:** Permitió obtener un procesamiento de la bibliografía necesaria para llevar a cabo la investigación.

Métodos empíricos:

1. **Entrevista:** Facilitó la obtención de información a través de personas responsables de desarrollar toda la integración de otros sistemas.

Los resultados esperados con esta investigación son:

1. Una aplicación estable que permita la administración de las estaciones de trabajo y los dispositivos instalados en las mismas, así como la generación de reportes según las necesidades del cliente.
2. Manuales de usuario para la aplicación web desarrollada.

El presente documento consta de tres capítulos, desarrollados a partir del estudio realizado de la

información bibliográfica e institucional existente. La descripción de los mismos se presenta a continuación:

- **Capítulo 1: Fundamentación teórica:** cuenta con el respaldo teórico de los temas tratados en el trabajo de investigación, necesarios para el correcto entendimiento de la solución propuesta. Presenta además el estudio realizado de los sistemas existentes, realizándose un análisis de la situación actual. Se presenta el ambiente de desarrollo necesario para el análisis y la implementación del módulo para la administración centralizada de dispositivos de hardware.

- **Capítulo 2: Análisis y diseño:** se describen las actividades del flujo de procesos a través de un modelo de dominio, el cual sirve de base para determinar lo que se va a desarrollar. Además se realiza el levantamiento de requisitos que contiene la definición de los requisitos funcionales y no funcionales del módulo, conjuntamente con los roles que intervienen. Se describen la arquitectura y los patrones que serán usados para el desarrollo del módulo, y se presentan las funcionalidades y el modelo de datos usado.

- **Capítulo 3: Implementación y prueba:** se lleva a cabo la descripción de las tareas de implementación generadas por cada historia de usuario realizadas en las tres iteraciones propuestas. Se realizan pruebas unitarias y de aceptación a la aplicación validando así la solución propuesta a través de los resultados obtenidos.

Capítulo 1 Fundamentación teórica

Introducción

En el presente capítulo se abordan los conceptos fundamentales y sistemas existentes relacionados con la administración centralizada de dispositivos de hardware. Se hace referencia a las tendencias y tecnologías actuales sobre metodologías de desarrollo y herramientas para el modelado de datos. Además se realiza un estudio de algunas tecnologías libres de desarrollo, lenguajes de programación y sistemas gestores de base de datos, con el objetivo de seleccionar los más indicados para el desarrollo de una solución de software con las características deseadas.

1.1 Conceptos fundamentales

Al profundizar en los sistemas existentes con la administración centralizada de dispositivos de hardware es importante destacar los diversos significados que se le acreditan a los mismos. Entre los conceptos fundamentales se encuentran “dispositivos de hardware”, y “administración centralizada de dispositivos”.

Dispositivos de hardware

Los dispositivos de hardware son aparatos o mecanismos que desarrollan determinadas acciones. Su nombre está vinculado a que dicho artificio está dispuesto para cumplir con su objetivo. (2)En la actualidad los dispositivos de hardware juegan un papel crucial en el desarrollo científico técnico, pues permiten a las estaciones de trabajos y a las aplicaciones que se ejecutan en ellas, integrarse en todas las áreas de la ciencia y la técnica.

Existen diversos dispositivos que posibilitan la realización de tareas diariamente, como es el caso de la impresora, escáner, micrófono, cámara digital, dispositivos de almacenamiento, entre otros. Coexisten con ellos otro grupo de dispositivos que son utilizados con frecuencia dentro de las soluciones empresariales y se encargan de automatizar tareas más complejas, ejemplo de estos son los lectores de huellas dactilares, tablas electrónicas para captar firmas, relojes para controlar la entrada y salida de personal, lectores de tarjetas, entre otros.

Administración centralizada de dispositivos

La administración es el proceso de planear, organizar, liderar y controlar el trabajo de una organización además de utilizar todos los recursos de la empresa para lograr objetivos organizacionales específicos. La administración de dispositivos, es la administración de todos los recursos del hardware disponibles, tanto los estándar que viene de fábricas, como las que se van agregando para hacer más poderosa o actualizar las computadoras. Todo dispositivo necesita presentarse al sistema operativo, agregando un pequeño programa que permite su uso.

La administración centralizada de los dispositivos hace posible establecer las políticas de seguridad, la configuración de las conexiones a los recursos, así como la gestión de los mismos. También permite la creación, inhabilitación y eliminación de cuentas de usuarios, dándole a estos los permisos necesarios para su desempeño laboral a través de una unidad central.

De acuerdo a los conceptos antes planteados los autores de la presente investigación definen como Sistemas para la administración centralizada de dispositivos del hardware como: sistemas que posibilitan controlar de forma centralizada los dispositivos de hardware, obteniendo reportes de uso de cada uno de ellos además de verificar las acciones realizadas por los usuarios.

1.2 Análisis de las soluciones existentes

Actualmente en el mundo existen sistemas que permiten la comunicación con dispositivos de hardware por lo que se hace necesario un estudio de los mismo con el objetivo de analizar la forma en cada uno de ellos administra dichos dispositivos; tal es el caso del sistema Axiom DMS y el Software de Administración y Enrolamiento Centralizado (MEMS).

En la Universidad de Ciencias Informáticas existen dos sistemas con características similares: Gestor de Recursos de Hardware y Software (XILEMA SGRHS) desarrollado en el Centro de Telemática de la Facultad 2 y el Device Grid Manager (DGM) v3.0 empleado actualmente para la interacción con los dispositivos de captura biométrica utilizados por el Sistema Único de Identificación Nacional (SUIN).

Resulta oportuno realizar un análisis de las soluciones mencionadas anteriormente con el propósito de evaluar su comportamiento ante la situación problemática planteada.

1.2.1 Axiom Device Management System (DMS)

El sistema Axiom DMS es una aplicación de Servidor, fabricado por Axiompass. Permite configurar la comunicación con terminales de registro y con los dispositivos conectados a ellas y cuenta con un mecanismo de consulta y registro de transacciones y accesos a través de dichas terminales. (3)

Axiom DMS puede correr bajo sistemas operativos como Windows XP, Windows Vista o Windows Server 2008 y utiliza una base de datos centralizada sobre Microsoft SQL Server 2000, Microsoft SQL Server 2005 o Microsoft SQL Server 2008.

El servidor Axiom DMS controla múltiples terminales que a su vez controlan dispositivos como pueden ser lectores biométricos, lectores de tarjeta, puertas, sirenas, luces, etc. La configuración de cada dispositivo es administrada por Axiom DMS, almacenada en la base de datos y transmitida a las terminales. Todas las transacciones y eventos generados por los dispositivos son recibidos por el servidor Axiom DMS y almacenados en la base de datos. A través de la red, clientes remotos pueden conectarse al servidor Axiom DMS para administrar o controlar dispositivos y terminales.

Algunas de las características del sistema Axiom DMS son:

- Controla un número prácticamente ilimitado de dispositivos.
- Administra la configuración y comportamiento de cada dispositivo y almacena dicha información en una base de datos centralizada.
- Cuenta con un módulo de diagnóstico en línea que permite ver el estado de los dispositivos de manera remota.
- Cuenta con un módulo de descarga simultánea de archivos e información a terminales.
- Cuenta con un servidor que permite a clientes autorizados monitorear el funcionamiento o interactuar con los dispositivos de manera remota.
- Cuenta con una interfaz gráfica y amigable.

Uno de los principales problemas del Axiom DMS es que cuenta con una licencia propietaria y con un alto costo monetario, además de estar altamente adaptado a los estándares del Sistema Operativo Windows, y no cuenta con ningún módulo para la interacción con dispositivos desde aplicaciones web en sus clientes.

1.2.2 Software de administración y enrolamiento centralizado (MEMS)

Es un software fabricado por Safran Morpho que ofrece la posibilidad de enrolar a las personas en una base de datos central a través de una estación de trabajo y luego replicar esta información a los distintos dispositivos conectados a la red. También es posible configurar de forma remota los dispositivos biométricos y recuperar centralizadamente la información de auditorías.

Para que este sistema pueda distribuir los patrones de identificación a los terminales y a la vez recuperar la información desde un punto central de administración, se requiere que los distintos terminales cuenten con una conexión a una red de comunicaciones de datos. El dispositivo se integra a la red de comunicaciones a través del protocolo estándar TCP/IP.

Entre las características más relevantes del sistema se encuentran:

- Interfaz gráfica de fácil uso.
- Opera en una estación de trabajo cliente con sistema operativo Windows.
- Base de datos central de huellas dactilares: Microsoft Access, SQL Server, Oracle.
- Consola de configuración, administración en red y reportes para múltiples terminales.
- Arquitectura Cliente/Servidor disponible con versión Microsoft Access, SQL Server y Oracle.
- Recuperación de trazas y generación de reportes.
- Funcionalidad de enrolamiento de huellas dactilares y registro de información básica del personal.
- Compresión de imagen y verificación de huellas.
- Control de accesos de acuerdo a zonas de tiempo.

Muy similar al Axiom DMS presenta la desventaja de no poderse integrar a las aplicaciones web, además de su alto precio en el mercado del software que oscila entre 1 000 y 2 000 libras esterlinas.

1.2.3 Gestor de Recursos de Hardware y Software (XILEMA SGRHS)

El sistema Gestor de Recursos de Hardware y Software desarrollado en el Centro de Telemática de la Facultad 2 es un sistema multiplataforma, utiliza una aplicación cliente-servidor cuyo objetivo principal es centralizar en el servidor la información del hardware y el software instalados en las estaciones clientes.

La aplicación cliente se encarga de obtener la información del hardware y software instalados y envía los datos recolectados al servidor, determinando si hubo cambios en el hardware o el software a partir del inventario obtenido anteriormente.

Facilita detectar cambios en dichos componentes, clasificarlos en Autorizados o No autorizados, según se haya establecido en la entidad, así como tomar acciones de control (apagar, hibernar, suspender, reiniciar, enviar notificación por correo) ante los cambios no autorizados. Posee una interfaz web que permite la visualización y administración de la información obtenida de las computadoras. El sistema agente así como el sistema servidor son propiedad exclusiva de la Universidad de las Ciencias Informáticas (UCI).

1.2.4 DGM V2.0

El Device Grid Manager (DGM) es una aplicación que permite el control e interacción centralizado de dispositivos de hardware en aplicaciones web. Esta aplicación permite al Sistema Único de Identificación Nacional de la República de Cuba (SUIN) la comunicación e interacción con los dispositivos de hardware instalados en las estaciones de trabajo para a partir de ellos obtener los datos necesarios para llevar a cabo los procesos de captación biométrica.

El sistema se compone por tres subsistemas fundamentales: el Servicio DGM para el manejo de local de dispositivos, el marco de trabajo JavaScript y el Sistema para el control centralizado de dispositivos. Estos sistemas trabajan como un todo, permitiendo la interacción y control de los dispositivos de hardware de una forma cómoda y elegante, basada en un desarrollo estandarizado que facilita la extensión de la aplicación sin grandes esfuerzos por parte de los desarrolladores.

Entre las características más relevantes del DGM v3.0 se tienen:

- Permite una integración de forma fácil y segura a cualquier aplicación o módulo, de forma tal que pueda ser usado en los distintos procesos de captación definidos en un sistema.
- Permite la incorporación de nuevos dispositivos en la medida que vayan surgiendo las necesidades para nuevos módulos, además acepta otros equipos con las mismas características que el actual pero de otros fabricantes, de manera que se puedan hacer reemplazos sin hacer cambios en las aplicaciones.

- Es totalmente configurable por lo cual se pueden definir los dispositivos habilitados sin necesidad de recompilar la aplicación aun siendo necesario usar dispositivos de otros fabricantes para realizar las mismas u otras operaciones.

1.3 Propuesta de solución

Los sistemas analizados cuentan con una serie de ventajas y desventajas las que han sido estudiadas, haciendo un aporte importante para sentar las bases del desarrollo de una nueva solución para la administración centralizada de dispositivos de hardware.

Sistemas	Sistema multiplataforma	Reportes de uso	Interacción desde aplicaciones web	Tecnologías libres
Axiom DMS		✓		
MEMS		✓		
DGM v 2.0		✓	✓	
XILEMA SGRHS	✓	✓	✓	✓

Tabla 1 Sistemas analizados y sus características. Fuente: Elaboración propia

En la tabla 1 se muestra una comparación entre los principales sistemas estudiados, atendiendo a 4 criterios: sistema multiplataforma, reportes de uso, interacción desde aplicaciones web y tecnologías libres. Estos criterios fueron escogidos para la comparación de los sistemas existentes ya que son funcionalidades que cumplirá el sistema de administración centralizada de dispositivos.

Sistema multiplataforma: Un sistema es multiplataforma cuando tiene la capacidad de funcionar en más de un sistema operativo tales como Windows, Unix, Linux, Mac OS, con similares características y sin que su funcionalidad varíe en exceso. (4)

Reportes de uso: Un reporte es un informe o documento que puede ser impreso, digital, etc. que pretende transmitir una información, aunque puede tener diversos objetivos. Los reportes de uso al cual se refiere es el informe de los dispositivos de hardware que se encuentran registrado en el sistema.

Interacción desde aplicaciones web: La interacción desde aplicaciones web, es cuando un sistema o un servicio es capaz de integrarse a una aplicación web correspondiente y a través de la misma visualizar las funcionalidades que brinda dicho sistema.

Las tecnologías libres son herramientas desarrollada en software libre el mismo respeta la libertad de los usuarios. Significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Es una cuestión de libertad, no de precio. (5)

Estos sistemas fueron desarrollados para lograr eficiencia y calidad en el marco al que fueron aplicados, pero muchos de ellos no constituyen una solución factible para Cuba, ya que son muy costosos y trabajan sobre software propietarios, lo cual entra en contradicción con las políticas de migración de software libre que se está llevando a cabo en el país. Sin embargo poseen ciertas características las cuales se tomarán en cuenta en esta investigación, como son interfaz amigable, la generación de reportes en varios formatos y filtros de búsqueda: por cantidad de memoria, sistema operativo, etc.

Utilizando las experiencias de los sistemas estudiados y adicionando nuevas funcionalidades se pretende realizar un nuevo sistema multiplataforma que permita la administración centralizada de dispositivos del hardware y la obtención de reportes de tiempo de uso por parte del usuario en varios formatos de los dispositivos, posibilitando supervisar y planificar su mantenimiento.

1.4 Metodologías para el desarrollo de software

El desarrollo de software no es una tarea sencilla, prueba de ello son las investigaciones que se realizan con el fin de obtener el éxito en este campo y lograr la calidad del producto que es el principal objetivo de las organizaciones. De ahí que los desarrolladores prestaran más atención a este tema y surgieran nuevas propuestas metodológicas con diferentes resultados en el proceso de desarrollo del software. (6).

Las metodologías para el desarrollo de software consisten en un conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Una metodología durante su ciclo de vida indica qué es lo que hay que obtener durante todo el proceso de desarrollo, es decir, cómo se obtienen los productos parciales y finales pero no cómo hacerlos. (7).

Estas metodologías se clasifican en dos grandes grupos: metodologías pesadas o tradicionales y metodologías ágiles o ligeras:

Las metodologías tradicionales están enfocadas en el control de proceso donde se establecen rigurosamente las actividades que se realizarán y los artefactos que se obtendrán. Estas metodologías han demostrado ser efectivas y necesarias en proyectos de larga duración pero han presentado problemas en proyectos donde el entorno es muy cambiante y en el cual se exige reducir el tiempo de desarrollo pero sin que afecte la calidad del producto.

Las metodologías ágiles están enfocadas en el factor humano donde se establece la colaboración con el cliente y el desarrollo incremental del software con iteraciones muy cortas. Se caracterizan por ser flexibles al cambio y por poder reducir el tiempo de desarrollo del software demostrando así que es más importante contar con un software funcional que una documentación excesiva. Estas metodologías han demostrado tener éxito en proyectos pequeños. (8)

A partir de lo analizado con anterioridad, teniendo en cuenta que el sistema a desarrollar no posee un gran número de funcionalidades a implementar y el equipo de desarrollo es pequeño, se decide enfocarse en el estudio de las metodologías ágiles por ser las idóneas para este tipo de proyecto. A continuación se detallan las características esenciales de las metodologías ágiles más destacadas en la actualidad.

1.4.1 Extreme Programming (XP)

XP es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y retroalimentación o reutilización del código desarrollado. Esta metodología se caracteriza por centrarse en fortalecer las relaciones interpersonales para el éxito del desarrollo de software, promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen ambiente de trabajo. XP se basa en la interacción continua entre el cliente y el equipo de desarrollo caracterizándose por mantener una conversación fluida, además de poseer soluciones implementadas simples y la capacidad de afrontar los cambios.

Debido a estas particularidades XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (6)

El ciclo de vida propuesto por la metodología se divide en las siguientes fases: (6)

- **Fase de exploración:** los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

- **Fase de planificación:** el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, además de las entregas relacionadas con éstas. El resultado de esta fase es un Plan Entregas.

- **Fase de iteraciones:** esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entregas está compuesto por las diferentes iteraciones. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto, escogiendo las historias de usuario que fuercen la creación de esta arquitectura, sin embargo, no siempre es posible ya que es el cliente quien decide que historias se implementarán en cada iteración. Al final de la última iteración el producto estará listo para entrar en producción.

- **Fase de puesta en producción:** en esta fase se entregan módulos funcionales y sin errores, y según los intereses del cliente el sistema puede ponerse en producción o no. No se realizan desarrollos funcionales, pero pueden ser necesarias tareas de ajuste.

- **Fase de mantenimiento:** mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para esto se requiere de tareas de soporte para el cliente. La velocidad de desarrollo puede disminuir después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

- **Fase de muerte del proyecto:** el cliente no tiene más historias para ser incluidas en el sistema, haciendo necesario que se satisfagan sus necesidades en otros aspectos tales como rendimiento y confiabilidad del mismo. Se genera la documentación final y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Algunas de las características fundamentales de XP son: (9)

- Pruebas unitarias: se basa en las pruebas realizadas a los principales procesos con el objetivo de detectar futuros errores.
- Re fabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Programación en pares: consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

A continuación se detallan las principales ventajas de la metodología XP: (8)

- Apropiada para entornos volátiles.
- Preparada para el cambio, lo que significa reducir su coste.
- Planificación más transparente para los clientes, conocen las fechas de entrega de las funcionalidades.
- Permite definir en cada iteración cuáles son los objetivos de la siguiente.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

Entre las desventajas de la metodología XP se pueden considerar las siguientes: (8)

- No se tiene la definición de costo y el tiempo de desarrollo.
- El sistema va creciendo después de cada entrega al cliente y no se puede saber con certeza si el cliente no necesitará una funcionalidad más.
- Se requiere de la presencia constante del usuario, lo cual en la realidad es muy difícil de lograr.

1.4.2 Proceso Unificado Ágil (AUP)

El Proceso Unificado Ágil (AUP) es una versión simplificada del Proceso Unificado de Rational (RUP). Describe de manera fácil la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos válidos para RUP. AUP se enfoca principalmente en la gestión de riesgos, haciendo la propuesta de que aquellos elementos con alto riesgo tengan prioridad en el proceso de desarrollo y sean tratados en etapas tempranas del mismo. El proceso establece un modelo más simple que el de RUP porque integra en una sola las disciplinas de modelado del negocio, requisitos y análisis y diseño, en el caso del resto de las disciplinas coinciden con las restantes de RUP. (10)

Las principales características de la metodología AUP son: (10)

- Abarca siete flujos de trabajos, cuatro de ingeniería y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente.
- El modelado agrupa los tres primeros flujos de RUP (Modelado del negocio, Requerimientos y Análisis y Diseño).
- Dispone de cuatro fases igual que RUP: Creación, Elaboración, Construcción y Transición.

La metodología AUP es ágil porque está basada en los principios ejemplificados a continuación: (10)

- Simplicidad: Todo se describe concisamente utilizando poca documentación.
- Agilidad: El ajuste a los valores y principios de La Alianza Ágil.
- Centrarse en actividades de alto valor: La atención se centra en las actividades que en realidad lo requieren, no en todo el proyecto.
- Facilidad de adaptar el producto que se obtiene de manera que satisfaga sus propias necesidades.

1.4.3 Scrum

La metodología Scrum es un modelo de referencia que define un conjunto de prácticas y roles que pueden tomarse como punto de partida para el proceso de desarrollo de software.

Esta metodología está indicada para proyectos con un alto grado de cambios en los requisitos. Su principal característica es que el desarrollo de software se realiza mediante iteraciones denominadas sprint, donde cada sprint representa un incremento del producto y las reuniones diarias que se llevan a cabo a lo largo del proyecto para la coordinación e integración de las actividades a desarrollar.

Aunque Scrum se enfoca en la gestión de procesos de desarrollo de software, puede ser utilizado en equipos de mantenimiento de software, o en una aproximación de gestión de programas: Scrum de Scrums. (11)

Las principales características de la metodología Scrum son las siguientes: (12)

- Permite la creación de equipos auto organizado impulsando la colocalización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.
- Posibilita que los clientes puedan cambiar de idea sobre lo que quieren y necesitan durante el desarrollo del proyecto.
- Maximiza la capacidad del equipo de realizar las entregas rápidamente y de respuesta a los requisitos emergentes.

1.5 Fundamentación de la metodología a utilizar

Después del análisis de cada metodología propuesta se decide utilizar la metodología XP pues para el desarrollo de la aplicación se cuenta con un equipo de proyecto pequeño y la programación se realiza en parejas. Además que es un desarrollo iterativo e incremental, interactúa con el usuario final y los requisitos pueden ser cambiados debido a que se pueden arrojar nuevas necesidades y funcionalidades durante el proceso de desarrollo del sistema. Además que XP permite la simplicidad de código, una mejor comunicación y una alta calidad en el producto en un mínimo período de tiempo.

1.6 Herramientas CASE.

Las herramientas CASE (Ingeniería de Software Asistida por Computadora) son aplicaciones informáticas dedicadas al aumento de la productividad en el desarrollo de software. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas, como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (13)

Entre las principales herramientas CASE utilizadas se encuentra Visual Paradigm, la cual esta detallada a continuación.

1.6.1 Visual Paradigm 8.0

Visual Paradigm 8.0 es una herramienta de modelado profesional que soporta el ciclo de vida completo del desarrollo de software. La herramienta de modelado UML (Lenguaje de Modelado Unificado) permite una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite modelar todos los tipos de diagramas de clases así como generar código inverso, código desde diagramas y documentación. Es fácil de instalar y actualizar además de ser compatible entre ediciones. Dentro de las principales características de la herramienta se encuentran: disponibilidad de múltiples versiones, de acuerdo a su necesidad, así como la capacidad de integrarse a los principales IDE (Entorno de Desarrollo Integrado). Además la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto. Es capaz de generar la documentación del proyecto en diferentes formatos como HTML o PDF y permite el control de versiones. (14)

Entre sus características podemos encontrar**:** (14)

- Producto de calidad.
- Soporta aplicaciones Web.
- Compatibilidad entre ediciones.
- Se integra con varias herramientas Java (Eclipse/IBM WebSphere, NetBeans IDE).

Ventajas de Visual Paradigm: (14)

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Generación de documentación en formatos HTML y PDF.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.
- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Dada la caracterización planteada anteriormente se decidió utilizar Visual Paradigm como herramienta CASE para la realización del modelo de datos, debido a su alta capacidad de integración con lenguajes de programación. Es una herramienta multiplataforma, disponible en varios idiomas además de ser fácil de instalar y actualizar. Facilita la organización de los diagramas generando la documentación del proyecto en varios formatos. Además de ser capaz de generar código, modelo de datos, entre otras funcionalidades útiles para el desarrollo de un producto.

1.7 Lenguajes de programación

Un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. Estos lenguajes pueden usarse para crear programas que controlen el comportamiento lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación. (15)

1.7.1 PHP

PHP (acrónimo recursivo de *PHP: Hypertext Pre-processor*) es un lenguaje de programación de propósito general y de código abierto para la creación de páginas webs dinámicas que permite la creación de aplicaciones con interfaz gráfica, conexión a servidores de base de datos como Oracle, MySQL, PostgresSQL y puede ser ejecutado en sistemas Windows, Linux y Mac OS. Su sintaxis recurre a C, Java y Perl, siendo así sencillo de aprender. Ofrece una solución simple y universal para las paginaciones dinámicas de la web de fácil programación. Su elegante diseño lo hace más fácil de mantener y ponerse al día en comparación con el código de otros lenguajes. Debido a su amplia distribución PHP está soportado por una gran comunidad de desarrolladores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. El código se mantiene actualizado continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP. (16)

La característica más importante de PHP es que permite combinar código HTML y código PHP en una misma página web. Otra característica interesante de PHP es que permite realizar inclusiones dentro de un script PHP, de esta forma una página se puede dividir en varios módulos PHP que se desarrollan en

forma independiente, además pueden desarrollarse componentes reusables en otras páginas o incluso en otros sitios.

Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, aun estando dirigido a alguna en particular, el programador puede aplicar en su trabajo cualquier técnica de programación o desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes mediante la utilización de marco de trabajo. La gran facilidad, ahorro de tiempo y código, que actualmente brindan los marcos de trabajos, es sin dudas un aspecto a considerar para llevar a cabo el desarrollo de una aplicación web.

1.7.2 Python

El lenguaje fue creado por Guido Van Rossum con el objetivo de cubrir la necesidad de un lenguaje orientado a objetos de uso sencillo que sirviese para tratar diversas tareas dentro de la programación. Durante varios años Python se estuvo desarrollando, ya en el 2000 disponía de un producto literalmente completo y un equipo de desarrollo con el que se había asociado incluso en proyectos empresariales. (17)

Éste es un lenguaje independiente de plataforma y orientado a objetos, bien preparado para realizar cualquier tipo de programa, ya sean aplicaciones Windows a servidores de red e inclusive, páginas web.

Es un lenguaje orientado a objeto, no necesariamente se necesita compilar el código fuente para poder ejecutarlo, permite mantener de forma sencilla la interacción con el sistema operativo y es muy adecuado para manipular archivos de texto. Es un lenguaje sencillo pero muy potente, no genera ejecutables, sino que es el encargado de ejecutar el código.

Python provee al producto varias ventajas entre ellas se encuentran (18):

- Rápido de desarrollar.
- Sencillez.
- Sus bibliotecas hacen gran parte del trabajo.
- Soporta varias bases de datos.

- Rápida curva de aprendizaje.
- Soporte a través de los módulos de gran cantidad de funcionalidades.
- Tiene aplicación para algunos tipos de desarrollo: juegos en 3D, análisis de datos, desarrollo web, investigaciones científicas.
- Se puede ajustar al modelo o estilo de programación que se desee: ya sea funcional u orientado a objetos.

Python está en pleno desarrollo, con él se pueden desarrollar todo tipo de programas que se ejecuten en cualquier máquina. El equipo de desarrollo de Python está trabajando de manera cada vez más organizada y cuentan con el apoyo de una comunidad que está creciendo rápidamente.

1.8 Fundamentación del lenguaje de programación a utilizar

Como resultado de la comparación realizada anteriormente entre PHP y Python se seleccionó PHP como el más idóneo para la programación de la aplicación web debido a las características que posee dentro de las cuales podemos mencionar que puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux y muchas variantes, soporta la mayoría de servidores web de hoy en día, incluyendo Apache. Además de tener los autores de la investigación un previo conocimiento del lenguaje y la existencia de la gran comunidad de PHP desarrollada en la Universidad de Ciencias Informáticas.

1.9 Marco de trabajo

Un marco de trabajo (framework) en el desarrollo de software, no es más que 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. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Un marco de trabajo representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Algunos de los marcos de trabajo que se analizaron para el desarrollo de aplicaciones web son: Symfony 2, Zend Framework y Yii Framework los cuales se detallan a continuación.

1.9.1 Symfony 2

Symfony es el marco de trabajo de PHP basado en la arquitectura MVC (Modelo-Vista-Controlador), su primera versión surgió por el año 2005 como un proyecto de software libre el cual se ha convertido en uno de los frameworks más populares de PHP que existe en la actualidad, su fundador y mayor impulsor el francés Fabien Potencier dedica aún la mayor parte de su tiempo a corregir él personalmente los problemas que puedan existir y que son identificado por la gran comunidad de la cual se rodea este framework. (19)

Este proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Ha sido probado en numerosos proyectos reales y es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas Unix, Linux, entre otras, como en plataformas Windows. (19)

A continuación se muestran algunas de sus características:

- Fácil de instalar y configurar en la mayoría de las plataformas.
- Independiente de un sistema gestor de bases datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de convenir en vez de configurar, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas de arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, permite la integración con librerías desarrolladas por terceros.

1.9.2 Zend framework

Zend framework (ZF) es un marco de trabajo de código abierto para desarrollar aplicaciones web y servicios web con PHP 5. ZF es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. (20)

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend Framework conforman un *framework* de aplicaciones web al combinarse. ZF ofrece una implementación MVC, una abstracción de base de datos, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. (20)

1.9.3 Yii framework

Yii es un marco de trabajo de código abierto utilizado para desarrollar todo tipo de aplicaciones web. Es orientado a objetos y posee un alto rendimiento basado en componentes. Yii viene con una colección de documentos oficiales, tales como un tutorial para desarrollar un simple blog, una guía que recoge la descripción de cada función y una referencia de clases que ofrece todos los detalles acerca de las propiedades, métodos y eventos. Yii se puede utilizar de forma gratuita para desarrollar cualquier aplicación Web de código abierto. En general, el contenido de la documentación Yii puede ser copiado, modificado y redistribuido siempre y cuando la nueva versión de subvenciones dé las mismas libertades a los demás y reconozca a los autores del artículo de la documentación utilizada Yii. (21)

1.10 Fundamentación del marco de trabajo a utilizar

Para la realización del proyecto se utiliza Symfony 2 por haber sido ideado para exprimir al límite todas las nuevas características de PHP y por eso ser uno de los frameworks de PHP con mejor rendimiento. Además de que gracias a la peculiaridad que posee, de tener su arquitectura interna totalmente desacoplada permite que se reemplacen o se eliminen fácilmente aquellas partes que no encajen con el proyecto a desarrollar, lo cual le permite adaptarse con facilidad a los nuevos cambios que puedan surgir durante el proceso de desarrollo del software. También existen módulos que se han desarrollado para ser integrados a Symfony 2, para realizar operaciones tales como: generación de reportes en formato .pdf; manipulación de seguridad y usuarios, gestión de envíos de correos electrónicos entre otros. Symfony 2 posee características que lo hacen único con respecto a otros frameworks, como el soporte HTTP, el contenedor de inyección de dependencias, el sistema de plantillas Twig y el sistema de bundles.

1.11 Entorno de Desarrollo Integrado (IDE)

Un Entorno Integrado de Desarrollo (IDE) es un sistema que facilita el trabajo del desarrollador de software, integrando sólidamente la edición orientada al lenguaje de programación, la compilación o interpretación, la depuración, las medidas de rendimiento, la incorporación de los fuentes a un sistema de control de fuentes, etc., normalmente de forma modular. (22).

1.11.1 Netbeans

Netbeans es un Entorno de Desarrollo Integrado (IDE) que permite crear aplicaciones de escritorio, aplicaciones web, entre otras. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. (23)

Las principales características del NetBeans son:

- Entorno de desarrollo multiplataforma, multilenguaje y amigable tanto para usuarios novatos como para profesionales.
- Disponible en muchos idiomas.
- Es de código abierto.

- Desarrollado por módulos. Brinda la posibilidad de agregar nuevos módulos para aumentar su funcionalidad.
- Cuenta con una amplia documentación y una gran comunidad de usuarios.
- Admite distintos tipos de lenguaje como PHP, C++ y Java.

1.11.2 Eclipse

Eclipse es un entorno de desarrollo integrado, de código abierto y multiplataforma. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. Eclipse puede considerarse actualmente como el IDE referencia en el mundo del software libre, con el inconveniente de funcionar mejor sobre una máquina virtual Java que no es libre. Eclipse es mucho más que un simple IDE, es toda una comunidad de desarrolladores de código libre, dedicados a la implementación de mejoras del entorno. (24)

1.11.3 Zend Studio

Zend Studio es un IDE destinado a desarrolladores profesionales. Es compatible con las plataformas Linux, Mac y Windows e integrado para el lenguaje de programación PHP. Consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración.

Entre sus características se encuentran: excelente completamiento de código, coloreado en la sintaxis del código, administración avanzada de proyectos, múltiples lenguajes, incorpora el Framework de Zend, PHP Documentor, manual de PHP. Integración con subversión, los navegadores, integración avanzada con FTP. Soporte para Web Services, PHP4, PHP5 y SQL. **(25)**

1.12 Fundamentación del Entorno de Desarrollo Integrado a utilizar

Se utiliza como entorno de desarrollo el NetBeans, debido a que es un es un proyecto de código abierto, soporta lenguajes dinámicos como PHP y Java Script, es multiplataforma, posee todas las herramientas necesarias para crear aplicaciones profesionales ya sean de escritorio, empresariales, web, móviles y otras, en diferentes lenguajes de programación además de que también tiene integración con el marco de trabajo Symfony2 del cual posee algunas bibliotecas y plugins que el programador puede agregarle a su aplicación. Además posee integración con la consola de PHP por lo que puede ejecutar comandos internos del mismo, los cuales son muy útiles en el trabajo con el marco de trabajo Symfony2. El NetBeans permite el completamiento de código, además de insertar funcionalidades definidas como los constructores, métodos get y set entre otros. También se pueden instalar plugins para realizar distintas operaciones internas como deshabilitar el escaneador de proyectos, bibliotecas para comandos de Symfony2 y muchos más.

1.13 Servidores Web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. El término también se emplea para referirse al ordenador que ejecuta el programa. (26).

A continuación se detallan los principales servidores web de código abierto:

1.13.1 Servidor Apache

El servidor HTTP Apache es un servidor web HTTP el cual se encuentra bajo la licencia de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1.

Apache es un servidor potente y flexible. Es altamente configurable y extensible con módulos de terceros, proporciona el código fuente completo y viene con una licencia sin restricciones, se ejecuta en Windows

2000 o superior, Netware 5.x o superior, OS / 2, y la mayoría de las versiones de Unix, así como varios sistemas operativos. Es una aplicación que está en constante desarrollo y alienta los comentarios de los usuarios a través de nuevas ideas, informes de errores y parches. (26)

A continuación se muestran las características que posee Apache y que la llevo al éxito en la inserción y utilización en ámbitos empresariales, tecnológicos y educativos:

- Fundamentalmente corre sobre una multitud de plataformas y sistemas operativos.
- Ofrece tecnología libre y de código abierto.
- Es un servidor Web configurable y de diseño modular, capaz de extender su funcionalidad y la calidad de sus servicios.
- Es posible configurar y personalizar cada uno de los mensajes de error que se pueden producir por la utilización del servidor.
- Otra particularidad propia de Apache, es que al ser tan popular y utilizado, es posible encontrar gran cantidad de documentos, ejemplos y ayuda en internet en todos los idiomas. (26)

1.13.2 Servidor Nginx

Nginx es el segundo más popular de los servidores web de código abierto actualmente. (27)Se trata de un servidor HTTP ligero, construido tomando el rendimiento en cuenta, en particular para manejar diez mil clientes al mismo tiempo. En lugar de utilizar hilos para tramitar las solicitudes, como los servidores tradicionales, Nginx utiliza una arquitectura impulsada por eventos asíncronos. Es más escalable y usa menos, y más cantidades de memoria. Además de las funciones básicas de HTTP, Nginx también soporta servidores virtuales basados en el nombre y basados en la IP, de mantenimiento de conexión y conexiones canalizado, y FLV streaming. También puede ser configurado de nuevo y actualizado en línea sin interrupción de la tramitación del cliente. Es licenciado bajo una licencia similar a BSD. Se ejecuta en UNIX, GNU/Linux, BSD, Mac OS X, Solaris y Windows. (28)

1.14 Fundamentación del servidor de aplicación a utilizar

Se seleccionó Apache 2.0 como servidor web pues es una herramienta modular, con gran cantidad de extensiones para diversas tecnologías y cuenta con abundante documentación. Además de que

implementa muchas características de uso frecuente, tales como: la configuración de las páginas protegidas con contraseña con un enorme número de usuarios autorizados, la capacidad de ofrecer respuestas personalizadas a los errores y problemas, entre otras. Es una tecnología libre cuya licencia permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

1.15 Sistemas de gestión de bases de datos (SGBD)

Los sistemas de gestión de bases de datos son un tipo de software dedicado a servir de interfaz entre las bases de datos, los usuarios y las aplicaciones que las utilizan y que permiten la creación y manipulación de los objetos y las propias bases de datos. Los principales objetivos de un Gestor de Base de Datos son ante todo evitar la redundancia, eliminando así la maleabilidad, y mejorar así los elementos de seguridad de los datos e intimidad. (29)

Además los sistemas de Gestión de Base de Datos presentan múltiples características las cuales cumplen con distintos objetivos: (29)

Abstracción de la información: Ahorran a los usuarios, detalles acerca del almacenamiento físico de los datos. No influye si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios Niveles de Abstracción.

Independencia: La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

Redundancia mínima: Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

Consistencia: En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

Seguridad: La información almacenada en una base de datos puede llegar a tener un gran valor, por lo que disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

Integridad: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

Respaldo y recuperación: Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Control de la concurrencia: En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información o almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

Tiempo de respuesta: Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en dar la información solicitada y en almacenar los cambios realizados.

Independientemente del tipo, los SGBD son programas informáticos y, por tanto, pueden ser de software libre o de propietario. (30) Dentro de estos dos grupos se analizará brevemente algunos de los SGBD Libres y algunos propietarios con versiones libres.

1.15.1 PostgreSQL

Es un potente SGBD relacional orientado a objetos es libre, multiusuario, centralizado y de propósito general, publicado bajo la licencia Berkeley Software Distribution (BSD) .Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*). Es ampliamente considerado como el sistema gestor de bases de datos de código abierto más avanzado del mundo; fue

pionero en muchos conceptos que estuvieron disponibles en algunos sistemas de bases de datos comerciales de alto calibre como por ejemplo: control de acceso simultáneo, gestión de transacciones, puntos de seguridad; fue uno de los primeros intentos en implementar un motor de bases de datos relacional. (31)

PostgreSQL ofrece sofisticadas características como: (32)

- Organiza los datos mediante un modelo objeto-relacional.
- Capaz de manejar procedimientos, rutinas complejas y reglas.
- Cuenta con una API sumamente flexible propia para el trabajo con varios lenguajes de programación y procedurales como C, C++, .NET, Bash, Delphi, PL/Java, PL/Perl, PL/Tcl, PL/pgSQL, PL/Ruby, PL/PHP, PL/Python, PL/Scheme y PL/R.
- Ofrece transacciones que permiten el paso entre dos estados consistentes manteniendo la integridad de los datos.
- Es altamente extensible, soporta operadores, funciones, métodos de acceso y tipos de datos declarados por el usuario; soporta además sobrecarga de operadores, sobrecarga de procedimientos, vistas, particionamiento de tablas y datos.
- Soporta integridad referencial, la cual es utilizada para garantizar la validez de la información dentro de las bases de datos.
- Usa una arquitectura cliente/servidor basada en un proceso por usuario; existe un proceso maestro que se ramifica para proporcionar conexiones adicionales por cada cliente que se intenta conectar a PostgreSQL.
- Tiene incorporado el mecanismo WAL (Write-Ahead Logging), que incrementa la confiabilidad de las bases de datos al registrar los cambios antes de ser escritos al disco; lo que asegura que, en caso de ocurrir un fallo crítico en las bases de datos, exista un registro de transacciones del cual se pueda restaurar.

El gestor de base de datos PostgreSQL es robusto y por ende hoy en día es el más usado con respecto a gestores libres existentes como: SQLite, MySQL en su versión libre, FireBird, DB2 Express-C y Apache Derby, ya que presenta una mayor escalabilidad y rendimiento bajo grandes cargas de trabajo. A continuación se reflejan las diferentes ventajas que este gestor ofrece: (33)

- Instalación ilimitada**:** no hay costo asociado a la licencia del software.
- Soporte: existe una gran comunidad de profesionales y empresas que ofrecen soporte a PostgreSQL, de la cual el Grupo Global de Desarrollo de PostgreSQL es la principal.
- Estabilidad y confiabilidad legendaria: miles de compañías reportan que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad.
- Multiplataforma: soporta alrededor de 34 plataformas incluyendo Linux y Unix en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows.
- Posee herramientas gráficas de diseño y administración de bases de datos: existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (DataArchitect).
- Soporta funciones con privilegios por usuario**:** que pueden definirse para ejecutarse con los derechos del usuario administrador o con los derechos de un usuario previamente definido, además retornan filas donde las salidas pueden tratarse como un conjunto de valores retornados por una consulta.
- Soporta bloques de código: que se ejecutan en el servidor y que pueden ser escritos en diferentes lenguajes de programación con la potencia que presenta cada uno.

PostgreSQL, tiene algunas cuestiones que pueden incidir negativamente en su funcionamiento satisfactorio, las más significativas son: (33)

- Consume muchos recursos y sobrecarga con facilidad el sistema, si no es configurado de manera óptima.
- Las filas de las tablas tienen como límite de tamaño 8Kb, se puede ampliar a 32Kb recompilándolo, pero con un costo añadido en el rendimiento.
- La velocidad de respuesta es un poco deficiente al gestionar bases de datos relativamente pequeñas, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes; lo que depende de la configuración del postgresql.conf.

1.15.2 Oracle

Oracle es un sistema de gestión de base de datos fabricado por Oracle Corporation, utiliza la arquitectura cliente/servidor. Se considera uno de los sistemas de bases de datos más completos. Es un sistema gestor de base de datos robusto, tiene muchas características que nos garantizan la seguridad e integridad de los datos; que las transacciones se ejecuten de forma correcta, sin causar inconsistencias; ayuda a administrar y almacenar grandes volúmenes de datos; estabilidad, escalabilidad y es multiplataforma.

Su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco; recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySQL o FireBird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux. (34)

Entre sus principales características podemos encontrar: (34)

Versatilidad: Por su cualidad de multiplataforma se pueden crear códigos en Java o en C++ utilizando dicha base de datos tanto en Windows como en Linux. Exporta los datos y los migra desde una plataforma a la otra sin causar problemas de integridad o pérdida de datos, migra con su propio software tanto de SQL Server como de MySQL funcionando sobre cualquier plataforma libre, por lo que es muy útil para todos los programadores.

Potencia: Ofrece un rendimiento mucho mayor que cualquier otra plataforma de Base de Datos. Permite al administrador asignar sus propias zonas de memoria a los datos y cualidades, se controla en todo momento; tanto el crecimiento como el rendimiento de los distintos esquemas que componen una BD sobre Oracle, aunque esto suponga un problema, ya que los administradores deben estar pendientes de su configuración para no sufrir fallos debido a algún problema de almacenamiento.

Seguridad: La seguridad de Oracle como sistema gestor de base de datos es alta, pues al no tener 100 porciento de integración con Windows lo hace invulnerable a los defectos que posee el sistema operativo, además de poseer un sistema de seguridad muy avanzado.

Complejidad: Complejo para los administradores que no estén familiarizados con las bases de datos, su alto rendimiento es directamente proporcional a su nivel de complejidad.

1.15.3 MySQL

MySQL es un Sistema de Gestión de Base de Datos relacional, multihilo y multiusuario con más de seis millones de instalaciones, actualmente es el gestor de bases de datos de código abierto más popular del mundo. (35)

MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Ofrece características como: (36)

Interioridades y portabilidad

- Escrito en C y en C++.
- Funciona en diferentes plataformas.
- Proporciona sistemas de almacenamiento transaccional y no transaccional.

Seguridad: Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

Escalabilidad y límites**:** Soporte a grandes bases de datos. Se permiten hasta 64 índices. Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

Localización**:** El servidor puede proporcionar mensajes de error a los clientes en muchos idiomas.

1.16 Fundamentación del Sistema Gestor de Bases de Datos (SGBD) a utilizar

Tras el estudio e investigación realizada sobre los gestores de base de datos libres, se toma como decisión realizar una comparación entre MySQL y PostgreSQL. Ambos servidores son soluciones comprobadas al paso del tiempo y que compiten fuertemente con la base de datos de software propietario.

MySQL ha sido durante mucho tiempo la más rápida pero es la que cuenta con menos funciones de los dos sistemas de bases de datos, mientras que PostgreSQL es un sistema de base de datos más denso, la cual a menudo se describe como la versión de código abierto de Oracle.

La tabla 2 muestra una comparación entre estos Sistemas Gestores de Bases de Datos en cuanto a características usadas por aplicaciones de mediana o gran envergadura, y para uso empresarial, lo que ilustrará como se encuentran estos SGBD en cuanto a la implementación de funcionalidades.

Características	MySQL	PostgreSQL
Máximo tamaño de la Base de Datos	Ilimitado	Ilimitado
Máximo tamaño de una Tabla	Ilimitado	Ilimitado
Máximo filas en una Tabla	Ilimitado	Ilimitado
Herencia entre tablas	No	**Si**
Dominios	No	**Si**
Particionado de Tablas	No	**Si**
Procedimientos almacenados de varios idiomas	No	**Si**
Crear tipos definidos por el usuario	No	**Si**
Crear operadores definidos por el usuario	No	**Si**
Monitoreo de Base De Datos	**Si**	**Si**

Tabla 2: Características de MySQL, PostgreSQL Fuente: Elaboración propia

La tabla anterior muestra que PostgreSQL en cuanto a funcionalidades es más completo que MySQL. Por la amplia gama de funcionalidades que implementa además es reconocido como uno de los mejores motores de bases de datos del mundo.

Después de realizado el estudio de las características más importantes y comparar algunos SGBD libres, se definió que el SGBD PostgreSQL cumple con todas las características necesarias para llevar a cabo la Aplicación web del DGM para la administración centralizada de dispositivos de hardware en ambientes multiplataforma.

1.17 Herramientas de Mapeo Objeto-Relacional (ORM)

El mapeo de objeto-relacional (ORM), es un modelo de programación que convierte datos entre el lenguaje de programación orientado a objetos y el sistema de base de datos relacional utilizado en el desarrollo de la aplicación. Esta técnica de programación se encarga de convertir los datos de los objetos que se crean en la aplicación en datos primitivos que se podrán almacenar en las tablas correspondientes de las bases de datos. Si luego se necesita ese objeto en la aplicación la técnica se encarga de recuperar los datos primitivos de la base de datos y construye el objeto. (37)

Las herramientas de mapeo objeto relacional se encargan no solo de manejar y plantear una solución a las diferencias expuestas, sino que además buscan reducir susceptiblemente el código necesario para llevar a cabo las operaciones de persistencia y recuperación de objetos. Proporcionan además interfaces más simples para el manejo de objetos a través de su propio lenguaje de consulta, y proveen al programador de configuraciones que le permiten optimizar los tiempos de respuesta en sus correspondientes aplicaciones.

Unas de las principales ventajas que ofrece la utilización de herramientas ORM son la facilidad y velocidad de uso, así como, la rapidez en el desarrollo. La mayoría de estas herramientas permiten la creación de un modelo libre de errores mediante los esquemas de tablas y relaciones. Si en futuros años se decide cambiar el motor de bases de datos, el modelo ORM garantiza que el cambio será más sencillo y no afectara al sistema.

Si las herramientas ORM son fáciles de utilizar pueden ser complejas, por lo que su correcta utilización y correcto funcionamiento necesita tiempo en el aprendizaje. Además todas las consultas que se hagan en

la base de datos son un poco lentas pues el sistema primeramente transforma en lenguaje propio, luego lee los registros y por ultimo crea los objetos. (37)

Entre las principales herramientas de ORM para el lenguaje PHP se encuentra Doctrine, Propel, Hibernate e Ibattis a continuación se detalla Doctrine que es el utilizado en la investigación:

1.17.1 Doctrine

Framework ORM para PHP en su versión 5.2, entre sus puntos fuertes se destaca su lenguaje DQL (Doctrine Query Language) que brinda la posibilidad de escribir consultas de base de datos. Este framework, para crear el modelo, nos da la alternativa de hacer una clase por tabla donde el programador puede especificar, mediante PHP, el tipo de datos almacenados, así como especificar relaciones y añadir funcionalidades extras a las clases autogeneradas. Una de las principales características es el bajo nivel de configuración que se necesita para empezar un proyecto.

1.18 Conclusiones parciales

En el presente capítulo se definieron los conceptos fundamentales de la investigación para lograr una mejor comprensión de la problemática planteada. Después de un estudio de los sistemas existentes que permiten la comunicación de dispositivos del hardware se evidenció que carecen de mecanismos que permiten su utilización desde aplicaciones web.

En esta primera fase se ha hecho un estudio profundo de las metodologías, lenguajes, tecnologías y herramientas, que ayudó a determinar y fundamentar, según las necesidades de la solución, la utilización de XP como metodología de desarrollo, Visual Paradigm como herramienta de modelado, PHP como lenguaje de programación, Symfony 2 como marco de trabajo, Netbeans como entorno de desarrollo, Apache2 como servidor web y PostgreSQL como SGBD a desarrollar.

Capítulo 2: Análisis y diseño

Introducción

En el presente capítulo se hace una descripción detallada de la propuesta de solución. Se exponen los artefactos generados propios de la metodología de desarrollo utilizada para la implementación de la solución que se propone tales como: los requerimientos funcionales y no funcionales con los que debe contar la aplicación, historias de usuario, tareas de ingeniería y tarjetas CRC (Contenido, Responsabilidad y Colaboración). Además también se describe la arquitectura del sistema, los patrones de diseño que fueron utilizados y el modelo de datos que fue generado para la solución propuesta.

2.1 Descripción del sistema propuesto

El Sistema para la administración centralizada de dispositivos brinda una interfaz visual en forma de aplicación web que permite al usuario gestionar cada uno de los dispositivos de la red así como ver en tiempo real el estado de estos. Permite además gestionar las localizaciones de los dispositivos, obtener reportes de uso y comportamiento de estos; el mismo se caracteriza por el fácil uso, permitiéndose usar prácticamente en cualquier escenario.

Esta aplicación web puede ser usada por todos los usuarios para generar los seriales y activar sus estaciones de trabajo sin tener necesidad de autenticarse en el sistema, accediendo de manera directa a los módulos de reportes y de activación de las estaciones de trabajo que le permitirán realizar dichas funcionalidades. Sin embargo los administradores del sistema requieren autenticarse iniciando su sesión a través de la pantalla de inicio de sesión. Si los datos introducidos por el usuario son correctos entonces se mostrará la interfaz principal de la aplicación donde podrán gestionar las agrupaciones lógicas, las agrupaciones físicas, los tipos de dispositivos, los dispositivos y los servicios, así como para desactivar desde ese sistema las estaciones de trabajo.

La aplicación está compuesta por 3 módulos que se resumen a continuación:

Gestionar Recursos

- Permite la gestión de todos los recursos que actúan en el sistema: agrupaciones lógicas, agrupaciones físicas, los tipos de dispositivos, los dispositivos y los servicios.

Estaciones de trabajo

- Permite la activación, desactivación y asignación o retiro de servicios y dispositivos de las estaciones de trabajo.

Reportes

- Permite la generación de los seriales para llevar a cabo la activación de las estaciones de trabajo, la obtención de reportes es varios formatos y obtener información sobre las estaciones de trabajo activadas a partir de una ubicación lógica.

2.2 Modelo de dominio

El Modelo de dominio es un artefacto que incorpora comportamientos y datos. Este modelo es la representación visual de las clases conceptuales u objetos del mundo real que reportan interés para la investigación o proyecto. Puede ser empleado en casos donde el negocio no comprende actividades demasiado cambiantes (38). En su construcción se modela un diagrama de clases compuesto por objetos del dominio o clases conceptuales, relaciones entre estas y los atributos que las componen.

Para un mayor conocimiento por parte del equipo de desarrollo quedan plasmados los conceptos fundamentales y una representación de sus relaciones en la Figura 1.

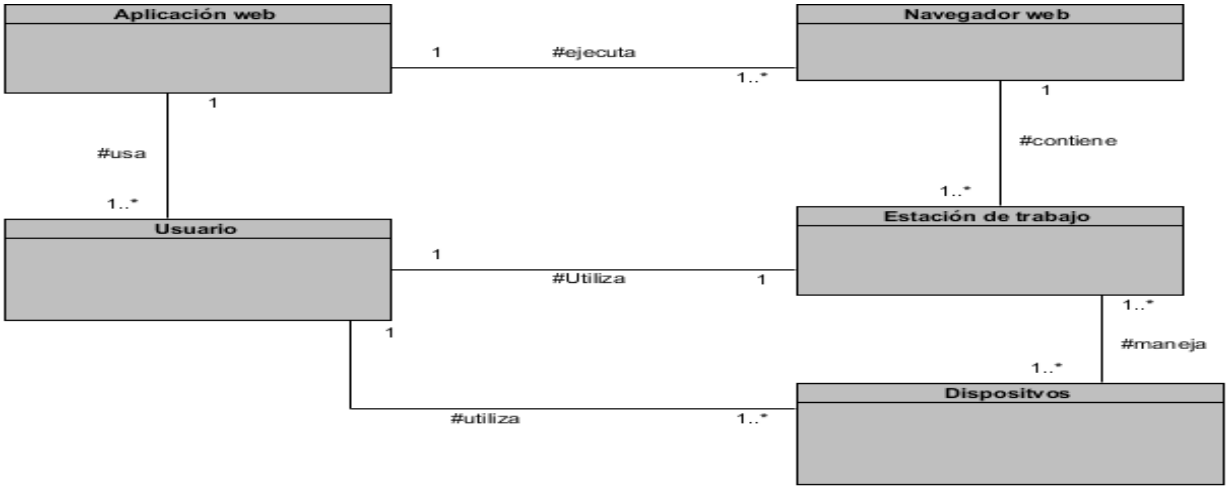

Figura 1 Modelo de Dominio. Fuente: Elaboración propia

Glosario de términos del modelo de dominio

En este acápite se explicará para un mejor entendimiento cada una de las clases que conforma el Modelo de dominio.

Usuario: Persona encargada de interactuar con la aplicación web y acceder a sus funcionalidades.

Estación de trabajo: Computadora destinada a ser usada por los usuarios para interactuar con las aplicaciones web mediante el navegador instalado en la misma, puede poseer un grupo de dispositivos conectados a ella.

Navegador web: Aplicación que se encuentra instalada en la estación de trabajo que permite la ejecución de una aplicación web.

Aplicación web: Contenido que se ejecuta dentro del navegador web, que está destinado para la interacción con el usuario a través de las funcionalidades que brindan, y que en determinados casos requieren el uso de dispositivos.

Dispositivos: Accesorios de hardware que se conectan a la estación de trabajo para dotarla de nuevas funcionalidades, que pueden ser utilizadas por aplicaciones, como pueden ser, capturar huellas, capturar

fotos e imprimir documentos. Estos dispositivos varían en dependencia de su marca, funcionalidades y distribuidor.

2.3 Requerimientos del sistema

Los requisitos son las condiciones o capacidades que deben ser alcanzadas o poseídas para que un sistema satisfaga las necesidades por las cuales fue creado. El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto. Los requisitos se pueden clasificar en funcionales y no funcionales. A continuación se enuncian los requisitos que fueron definidos para llevar a cabo el desarrollo de la solución propuesta.

2.3.1 Especificación de los requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir y no alteran la funcionalidad del sistema, esto quiere decir que los requisitos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen. La obtención de los requisitos funcionales constituye la actividad más importante dentro del modelación del negocio. Teniendo en cuenta las disposiciones para las que se creará el sistema fueron definidos los siguientes requisitos funcionales:

Módulo Gestionar Recursos

RF1. Gestionar agrupaciones lógicas.

RF 1.1 Mostrar listado de las agrupaciones lógicas existentes.
RF 1.2 Adicionar una agrupación lógica.
RF 1.3 Modificar una agrupación lógica.
RF 1.4 Eliminar una agrupación lógica.
RF 1.5 Actualizar listado de agrupaciones lógicas existentes.

RF2. Mostrar los datos de una agrupación lógica dado su identificador.
RF3. Gestionar agrupaciones físicas.

RF3.1 Mostrar listado de las agrupaciones lógicas existentes.

RF3.2 Adicionar una agrupación física.

RF 3.3 Modificar una agrupación física

RF 3.4 Eliminar una agrupación física.

RF4. Gestionar tipos de dispositivos.

RF 4.1 Mostrar listado de tipos de dispositivos existentes.

RF 4.2 Adicionar un tipo de dispositivo.

RF 4.3 Modificar un tipo de dispositivo.

RF 4.4 Eliminar un tipo de dispositivo.

RF5.Gestionar dispositivos.

RF 5.1 Mostrar listado de dispositivos existentes.

RF 5.2 Adicionar un dispositivo.

RF 5.3 Modificar un dispositivo.

RF 5.4 Eliminar un dispositivo.

RF6 .Gestionar servicios.

RF 6.1 Mostrar listado de servicios existentes.

RF 6.2 Adicionar un servicio.

RF 6.3 Eliminar un servicio.

RF 6.4 Modificar un dispositivo.

Módulo Estaciones de trabajo

RF7.Activar estación de trabajo.

RF 7.1 Registrar el serial de activación.

RF 7.2 Registrar el nombre del dispositivo y el tipo de dispositivo.

RF 7.3 Visualizar la estructura en la que se encuentra la ubicación física.

RF 7.4 Seleccionar la estación de trabajo según su ubicación física.

RF 7.5 Mostrar al usuario las especificaciones técnicas del dispositivo que está activando.

RF 7.6 Mostrar mensaje de confirmación en caso que se active el dispositivo correctamente.

RF 7.7 Mostrar mensaje de error en caso de ocurrir cualquier error mediante la activación del dispositivo.

RF8.Desactivar estación de trabajo.

RF 8.1 Mostrar listado de agrupaciones lógicas.

RF 8.2 Seleccionar la estación de trabajo según su ubicación física.

RF 8.3 Desactivar estación de trabajo seleccionada.

RF 9.Asignar servicios a las estaciones de trabajo según la ubicación lógica.

RF 9.1 Seleccionar la ubicación lógica.

RF10.Asignar dispositivos a las estaciones de trabajo según la ubicación lógica.

RF 10.1 Seleccionar la ubicación lógica.

Módulo Reportes

RF11.Generar reportes

RF 11.1 Mostrar listado de las de estaciones de trabajo activadas*.*

RF 11.2 Mostrar listado de los seriales de activación generados.

RF 11.3 Mostrar reporte de tiempo de uso de los dispositivos.

RF 11.4 Exportar reportes de usos en varios formatos.

RF12. Conocer estado del ordenador (encendido/apagado).

RF 12.1 Seleccionar la ubicación lógica.

RF 12.2 Mostrar listado de las estaciones de trabajo activadas de la ubicación lógica seleccionada.

RF 12.3 Mostrar el estado de la pc (encendida/apagado) de la seleccionada por la ubicación lógica.

2.3.2 Especificación de los requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. En muchos casos, estos requerimientos son fundamentales en el éxito del producto y generalmente están vinculados a requerimientos funcionales, además corresponden a aspectos tales como la disponibilidad, soporte, seguridad, apariencia e interfaz externa. A continuación se presenta el conjunto de requisitos no funcionales del sistema.

Requisitos de usabilidad

- El software debe permitir la integración del servicio local para el manejo de dispositivos, conformando así el Device Grid Manager.

Requisitos de soporte

- El sistema debe poseer un Manual de Usuario.

Requisitos de diseño e implementación

- Lenguaje de programación: PHP 5.5
- IDE: Netbeans 8.0
- Para el Modelado de UML se utiliza: Visual Paradigm 8.0
- Gestor de base de datos: Apache 2.2
- Gestor de base de datos: PostgreSQL 9.1

Requisitos de software

- El sistema debe ejecutarse en el sistema operativo GNU/Linux preferentemente Nova o Windows XP o superior.
- El sistema gestor de bases de datos, será PostgreSQL 9.1.
- Navegador Web: Mozilla Firefox 28.0 o versiones superiores, Chrome 33.0 o versiones superiores.

Requisitos de hardware

- Las estaciones de trabajo cliente deben tener como mínimo 128 Mb de RAM y procesador Pentium IV o superior.
- El servidor para la instalación del Sistema para la administración centralizada de dispositivos debe tener como mínimo 1 Gb de RAM, 20 GB de disco duro disponible para el almacenamiento de la Base de datos y procesador superior 2.40 GHZ.

2.4 Fase de exploración. Definición

El ciclo de vida de XP enfatiza en el carácter interactivo e incremental del desarrollo. Una iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas, que en el caso de XP corresponden a un conjunto de historias de usuarios (39).

La metodología de desarrollo XP comienza con la fase de exploración, en esta fase los clientes plantean a grandes rasgos las historias de usuario mediante un proceso de identificación que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología (39).

2.4.1 Historias de usuarios

Las historias de usuario (HU) son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (40)

Las HU conducen al proceso de creación de las pruebas de aceptación, las cuales servirán para verificar que estas historias se han implementado correctamente. Otra de sus características es que solamente proporcionan los detalles sobre la estimación del riesgo y cuánto tiempo conllevará su implementación.

Como resultado del trabajo realizado durante la fase de exploración se identificaron un total de 12 historias de usuario, a continuación se muestra una de ellas, el resto se encuentra en el Anexo 1.

Historia de Usuario	
Número: 1	**Usuario:** Administrador
Nombre historia: Gestionar agrupaciones lógicas	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1 semana	**Iteración asignada:**1
Programador responsable: Lien Morell Baro-Yasiel Lora Rodríguez	
Descripción: El sistema debe mostrar un listado de las agrupaciones lógicas existentes, permitir al usuario seleccionar la cantidad de información a mostrar en la página teniendo en cuenta las opciones que brinda (adicionar, eliminar y modificar). Además de actualizar listado de agrupaciones lógicas existentes después de la ejecución de cualquier acción (adicionar, modificar, eliminar). Mostrar un mensaje de confirmación después de haber ejecutado cualquier acción sobre una agrupación lógica y notificar en caso de errores a través de un mensaje de error.	

Observaciones:

Tabla 3 HU1 Gestionar agrupaciones lógicas.

2.4.2 Definición de los actores

Determinar a quién está dirigido un sistema informático es una de las premisas durante el desarrollo de cualquier producto. Para esta aplicación se identificaron los siguientes roles de usuarios: el administrador y el usuario funcionario. A continuación se muestra una tabla con las descripciones de cada uno de los roles.

Roles	Objetivo
Administrador	Encargado de gestionar toda la información referente a los puestos de trabajo y dispositivos.
Funcionario	Tiene acceso solamente a las siguientes acciones: “Activar estación de trabajo” y consultar el reporte: “Generar seriales de activación”.

Tabla 4 Roles de usuarios

2.5 Fase de Planificación. Definición

La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. Es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas. (39)

La planificación no debe ser estricta puesto que hay muchas variables en juego, debe ser flexible para poder adaptarse a los cambios que puedan surgir. Una buena estrategia es hacer planificaciones detalladas para unas pocas semanas y planificaciones mucho más abiertas para unos pocos meses.

2.5.1. Estimación del esfuerzo por historia de usuario

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Este se expresa utilizando como medida el punto. Un punto se considera como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción. (39)

El tiempo de implementación de una historia de usuario generalmente es de uno a tres puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

Para el desarrollo satisfactorio de la solución propuesta, se realizó una estimación de esfuerzo para cada una de las historias de usuario, arrojando los siguientes resultados:

No	Historias de usuarios	Puntos de estimación
1	Gestionar agrupaciones lógicas	1
2	Mostrar los datos de una agrupación lógica dado su identificador	1
3	Gestionar agrupaciones físicas	1

4	Gestionar tipos de dispositivos.	1
5	Gestionar dispositivos	1
6	Gestionar servicios	2
7	Activar estación de trabajo	1
8	Desactivar estación de trabajo	1
9	Asignar servicios a las estaciones de trabajo según la ubicación lógica	1
10	Asignar dispositivos a las estaciones de trabajo según la ubicación lógica	1
11	Generar reportes	1
12	Conocer estado del ordenador	2
Total		

Tabla 5 Plan de estimación de esfuerzo por historias de usuario

2.5.2 Plan de Iteraciones

En el plan de liberaciones se establece cuántas iteraciones serán necesarias realizar sobre el sistema para su consecución. Sin embargo, se precisa establecer el contenido de trabajo para todas y cada una de ellas y es aquí donde se establece el plan de iteraciones, regulando la cantidad de historias de usuario a implementar dentro del rango establecido por la estimación efectuada.

Tomando como referencia los aspectos antes tratados la aplicación que se pretende construir se desarrollará en 3 iteraciones, explicadas más detalladamente a continuación:

Iteración 1

La iteración tiene como finalidad implementar las historias de usuario que se consideraron más necesarias atendiendo a su relevancia e impacto para el negocio. Se da respuesta a las funcionalidades pertenecientes al módulo Gestionar Recursos.

Iteración 2

En esta iteración se realizan todas las historias de usuarios relacionadas con el módulo Estaciones de trabajo. Además, se corregirán errores o inconformidades con las HU implementadas en la iteración anterior.

Iteración 3

En esta iteración se realiza la historia de usuario relacionada con los reportes que se generan en el sistema. De esta manera se obtiene la versión 1.0 del producto final.

2.5.3 Plan de duración de las iteraciones

Para una mayor organización del trabajo como lo plantea el ciclo de vida de XP se crea un plan de duración de las iteraciones, en este caso se realizaría un solo plan ya que existe un único equipo de desarrolladores.

Este plan se realiza con el objetivo de reflejar cuáles serán las historias de usuario que se implementarán en cada una de las iteraciones, así como el tiempo destinado a cada una de ellas y el orden en que se implementarán, lo que ayuda a obtener una idea general del tiempo que durará la confección total del sistema.

Iteraciones	Historias de usuario	Duración
Iteración 1	Gestionar agrupaciones lógicas	7 semanas
	Mostrar los datos de una agrupación lógica dado su identificador	
	Gestionar agrupaciones físicas	
	Gestionar tipos de dispositivos	
	Gestionar dispositivos	
	Gestionar servicios	
	Activar estación de trabajo	

Iteración 2	Desactivar estación de trabajo	4 semanas
	Asignar servicios a las estaciones de trabajo según la ubicación lógica	
	Asignar dispositivos a las estaciones de trabajo según la ubicación lógica	
Iteración 3	Generar reportes	3 semanas
	Conocer estado del ordenador	

Tabla 6 Plan de duración de las iteraciones

2.5.4 Plan de entrega

El objetivo de este plan es producir rápidamente versiones de la aplicación que sean operativas, aunque estas no cuenten con toda la funcionalidad pretendida, pero sí que constituyan un resultado de valor para el negocio. El plan de entrega que se muestra a continuación propone el tiempo aproximado de entrega de versiones de la aplicación implementadas durante cada una de las iteraciones.

Módulo	Primera Iteración 19 de enero- 6 de marzo	Segunda Iteración 9 de marzo-3 de abril	Tercera Iteración 6 de abril-24 de abril
Versión	0.1	0.2	1.0

Tabla 7 Plan de entrega

2.6 Fase de Diseño

El diseño es fundamental, a diferencia de otras metodologías se realiza durante todo el tiempo de vida del proyecto por lo que se establecen los mecanismos, para que este sea revisado y mejorado continuamente según se van añadiendo funcionalidades al mismo.

XP establece prácticas especializadas que inciden directamente en la realización del diseño para lograr un sistema robusto y reutilizable tratando de mantener su simplicidad, es decir, crear un diseño evolutivo que

se va mejorando incrementalmente y que permite hacer entregas pequeñas y frecuentes de valor para el cliente. A la hora de darle cumplimiento a la actividad de diseñar, XP no especifica ninguna técnica de modelado. Pueden utilizarse indistintamente sencillos esquemas en una pizarra, diagramas de clases utilizando UML o tarjetas CRC (Clase, Responsabilidad y Colaboración) siempre que sean útiles, tributen a la comprensión y no requieran mucho tiempo en su creación.

2.6.1 Tarjetas CRC

Para el diseño de las aplicaciones, la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración).

Un modelo CRC es realmente una colección de tarjetas índice estándar que representan clases. Las tarjetas están divididas en tres secciones. A lo largo de la cabecera de la tarjeta usted escribe el nombre de la clase. En el cuerpo se listan las responsabilidades de la clase a la izquierda y a la derecha los colaboradores. (41)

Como resultado del trabajo realizado durante la fase de diseño se obtuvieron un total de 14 tarjetas CRC, a continuación se muestran algunas de ellas, resto se encuentra en el Anexo 2.

Clase: **Agrupacionfisica**	
Responsabilidades: **Permite devolver el valor de cualquier atributo de una agrupación física.** **Permite cambiarle el valor inicial de uno a varios atributos.** **Permite mostrar un listado de las agrupaciones físicas existentes.**	**Colaboradores**

Tarjeta CRC 3. Clase Agrupacion_fisica

Clase: **AgrupacionfisicaController**

Responsabilidades:	**Colaboradores**
Adicionar una agrupación física. **Eliminar una agrupación física.** **Modificar una agrupación física.** **Actualizar listado de las agrupaciones físicas.**	Agrupacionfisica

Tarjeta CRC 4. Clase AgrupacionfisicaController

2.6.2 Modelado de datos

Un modelo de datos es la representación abstracta de los datos en un sistema gestor de base de datos. Básicamente el modelo de datos está formado por tres elementos fundamentales que son:

- Objetos (entidades que existen y se manipulan).
- Atributos (Características básicas de estos objetos).
- Relaciones (forma en que se enlazan los distintos objetos entre sí).

En la siguiente figura, se muestra el modelo de datos del sistema. Este diagrama está compuesto por las entidades que permiten almacenar los datos de las agrupaciones lógicas y físicas, las entidades que almacenan los datos de los distintos dispositivos que interactúan con el sistema, los servicios y las entidades que representan los nomencladores del sistema. En el Anexo se encuentra la descripción de sus entidades.

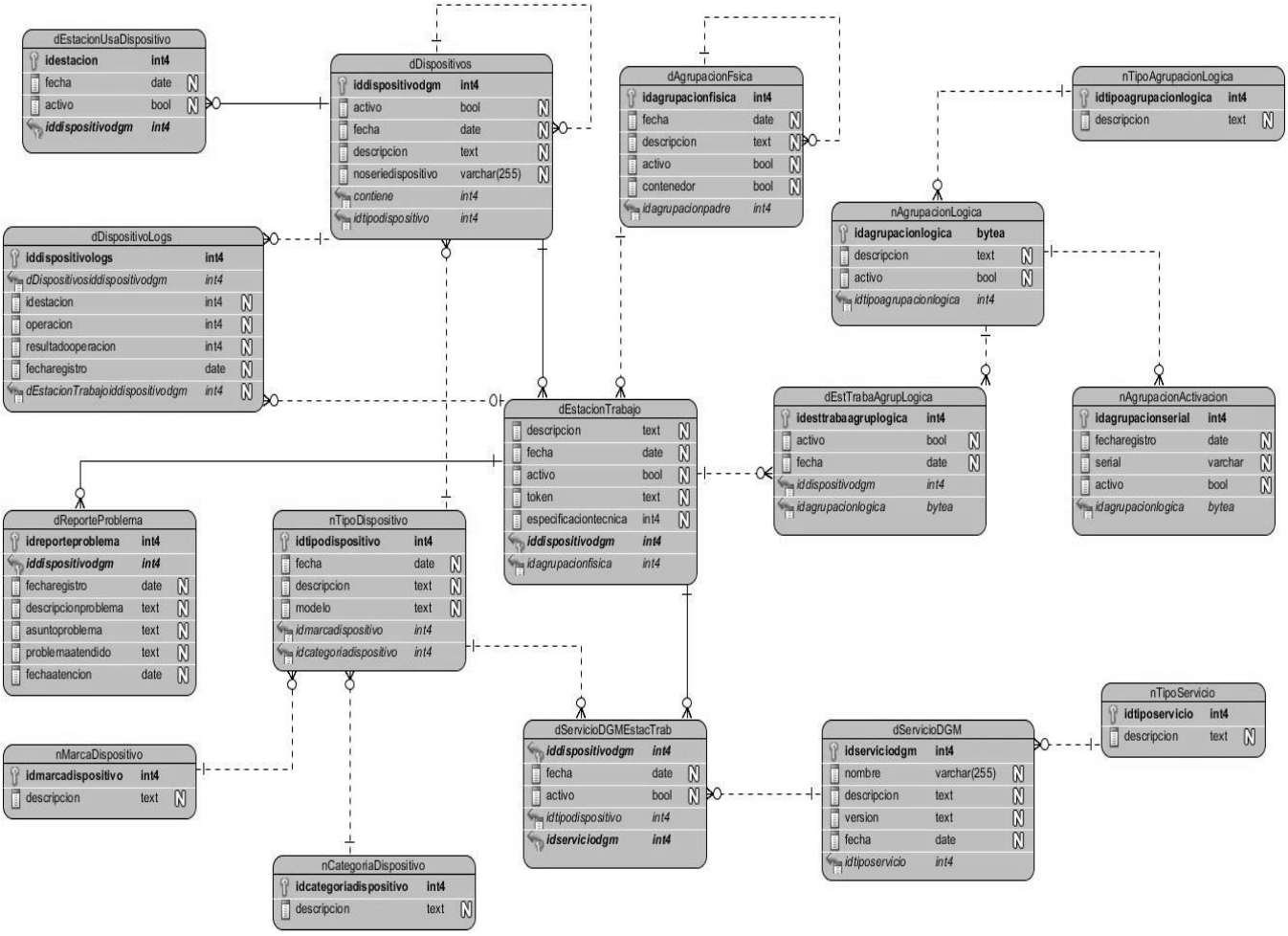

Figura 2 Modelo de datos del sistema DGM Free. Fuente: Elaboración propia.

2.7 Arquitectura del sistema

La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los principales componentes del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. Establece los fundamentos para que analistas, diseñadores, programadores y otros roles, trabajen en una línea común que permita alcanzar los objetivos del sistema, cubriendo todas las necesidades.

Se utilizó la arquitectura n-capas, la misma se basa en una distribución jerárquica de roles y responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de interacción con otras capas, y las responsabilidades la funcionalidad que implementan. El sistema se divide en las siguientes capas:

- **Capa Presentación:** es la encargada de comunicar al usuario con la aplicación, está compuesta por las interfaces de usuario y los componentes necesarios para su correcto funcionamiento. Se encuentra representada por la aplicación web y tiene interacción directa con la capa de negocio. También es conocida como interfaz gráfica y tiene la característica de ser entendible y fácil de usar por parte del usuario.

- **Capa del Negocio**: Es donde residen los programas que se ejecutan durante la ejecución de la aplicación. Esta capa se comunica con la de presentación, para recibir las peticiones y presentar los resultados, y con la de acceso a datos, para interactuar con la base de datos, consultando y modificando sus informaciones. En la misma se encuentran las clases del negocio que no son más que aquellas que permiten la gestión de las agrupaciones lógicas, físicas, etc.

- **Capa de Acceso a Datos**: Es donde residen los datos y la encargada de acceder y modificar a los mismos. Está formada por un sistema de gestor de bases de datos que realizan todo el almacenamiento de informaciones y que reciben solicitudes de almacenamiento o recuperación de datos desde la capa de negocio.

El estilo arquitectónico propuesto es el Modelo-Vista-Controlador (MVC) ya que separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones además viene integrado al Symfony2 en las capas de presentación y negocio. Cuando el usuario solicita alguna acción en el sistema, el sistema de enrutamiento determina qué controlador está asociado con la página solicitada por el usuario, Symfony2 ejecuta el controlador asociado a dicha página que el mismo solicita al modelo los datos correspondientes a la página solicitada. Con los datos devueltos por el modelo, el controlador solicita a la vista que cree una página mediante una plantilla y que inserte los datos del modelo. El controlador entrega al servidor la página creada por la vista y se le devuelve al usuario la página correspondiente a la acción seleccionada. En la figura 3 se muestra su funcionamiento:

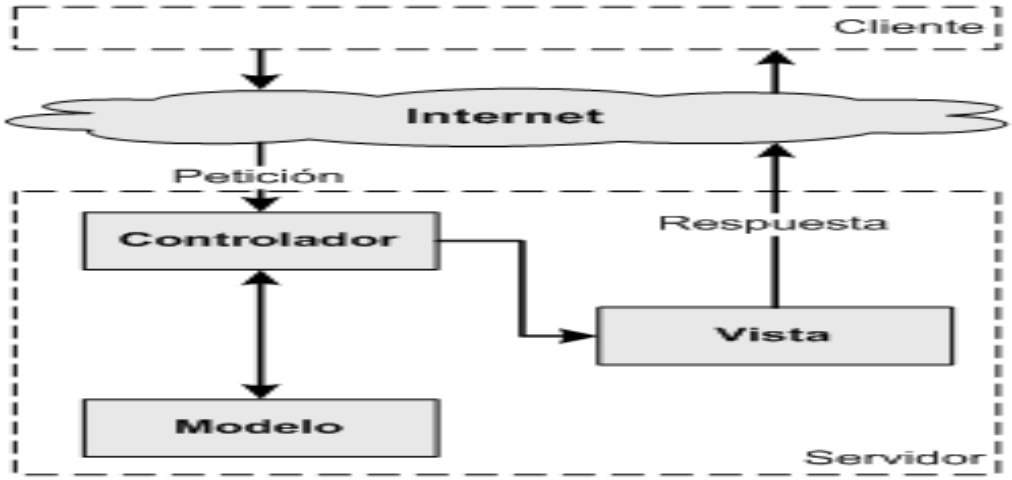

Figura 3 Patrón arquitectónico MVC (Modelo Vista Controlador)

2.8 Patrones de diseño utilizados

Son aquellos que expresan esquemas para definir estructuras de diseño o las relaciones que existen entre ellos a partir de las cuales se pueden construir sistemas de software. Proponen una forma de reutilizar experiencias de los desarrolladores, para ello clasifican y describen formas de solucionar problemas frecuentes durante el desarrollo de software. Estos se aplican a un elemento específico como un agregado de componentes para resolver un determinado problema de diseño, relaciones entre los componentes o los mecanismos para efectuar la comunicación componente a componente. (42)

Dentro de los patrones de diseño se encuentran dos grupos fundamentales conocidos por Patrones Generales de Asignación de Responsabilidades de Software (GRASP por sus siglas en inglés) y Banda de los Cuatro (GOF por sus siglas en inglés). Los Patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. De los patrones GRASP en la propuesta de solución se utilizaron los siguientes:

2.8.1 Patrón Experto

Permite asignar una responsabilidad al experto en información o sea a la clase que cuenta con la información necesaria para cumplir una determinada responsabilidad. Symfony2 utiliza este patrón con la inclusión de Doctrine para el mapeo de bases de datos. Se utiliza específicamente para crear una capa de

abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases entidades con todas las funcionalidades comunes (GET, SET y el constructor de la entidad); las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla asociada.

Beneficios del patrón Experto:

- Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para realizar todas las peticiones. Esto posibilita tener sistemas de fácil mantenimiento.
- El comportamiento se distribuye entre las clases que cuentan con la información requerida para cumplir con la responsabilidad asignada.

Este patrón se utilizó en la clase Agrupacionfisica. Un ejemplo se puede apreciar en la figura 4.

```
/**
 * @var \Agrupacionfisica
 *
 * @ORM\ManyToOne(targetEntity="Agrupacionfisica")
 * @ORM\JoinColumns({
 *   @ORM\JoinColumn(name="idagrupacionfisica", referencedColumnName="id")
 * })
 */
private $idagrupacionfisica;


/**
 * @var string
 *
 * @ORM\Column(name="ip", type="string", nullable=true)
 */
private $ip;
```

Figura 4: Ejemplo de patrón experto.

2.8.2 Patrón Creador

Su implementación permite identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Dicho de otra manera este patrón plantea que se debe asignar a una clase A la responsabilidad de crear una instancia de una clase B. Como la creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos es importante el uso de este patrón para guiar la asignación de responsabilidades relacionadas con la creación de objetos. Este patrón es utilizado

en la implementación de las clases controladoras donde se encuentran las acciones definidas para el sistema. En dichas acciones se crean los objetos de las clases que representan las entidades.

Beneficios del patrón Creador

- Se crean menos dependencias y existe mayor posibilidad de reutilización de código.

Un ejemplo se puede apreciar en la figura 5.

```
    public function createAction(Request $request)
    {
        $entity = new Agrupacionactivacion();
        $form = $this->createCreateForm($entity);
        $form->handleRequest($request);

        if ($form->isValid()) {
            $em = $this->getDoctrine()->getManager();
            $em->persist($entity);
            $em->flush();
$this->get('session')->getFlashBag()->add('info',
'La Agrupacion activacion ha sido creada'
);
            return $this->redirect($this->generateUrl('agrupacionactivacion_show', array('id' => $entity->getId())));
        }

        return $this->render('HardAdminBundle:Agrupacionactivacion:new.html.twig', array(
            'entity' => $entity,
            'form'   => $form->createView(),
        ));
    }
```

Figura 5: Ejemplo de patrón creador

2.8.3 Patrón Controlador

El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método encargado de la ejecución. Se manifiesta en todo el sistema debido a que cada uno de los eventos generados por el usuario son atendidos por el archivo routing.yml que es el encargado de redirigir la petición al método de una clase controladora que realice las operaciones solicitadas, pero siempre manteniendo las clases controladoras sin sobrecarga, es decir siempre manteniendo la alta cohesión.

2.8.4 Patrón Alta cohesión

La alta cohesión hace referencia a cuanta responsabilidad tiene una determinada clase controladora, o sea una clase debe tener responsabilidades moderadas. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que no realicen un trabajo enorme. Symfony2 permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Se puede observar en el sistema, ya que cada clase controladora maneja solamente las responsabilidades correspondientes a las entidades con las que se relaciona, además para cada vista existe una página controladora encargada de manejar sus solicitudes.

Beneficios del patrón Alta cohesión:

- Mejoran la claridad y la facilidad con que se entiende el diseño.
- Se simplifican el mantenimiento y las mejoras en funcionalidad.

2.8.5 Patrón Bajo acoplamiento

Este patrón plantea que se debe asignar las responsabilidades de forma tal que las clases dependan del menor número de clases que sea posible. Este patrón resuelve el problema de cómo dar soporte a una dependencia escasa y a un aumento de la reutilización. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas.

En Symfony2 las clases controladoras heredan de la clase Controller alcanzando de esta manera un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, y no tienen asociaciones con las de la vista o el controlador, lo que proporciona bajas dependencias.

Beneficios del patrón Bajo acoplamiento:

- Fáciles de entender por separado.
- Fáciles de reutilizar.

Los patrones GoF (del inglés Gang-of-Four) más conocidos como “Pandilla de los Cuatro” son utilizados básicamente para ocultarles a los usuarios la complejidad de un sistema, mostrándole solamente lo que necesita ver (43). Del patrón GOF se utilizó el siguiente:

2.8.6 Patrón Decorador

Añade funcionalidad a una clase dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. Tiene como ventaja que aporta una mayor flexibilidad que la herencia estática, permitiendo, entre otras cosas, añadir una funcionalidad dos o más veces, evita concentrar en lo alto de la jerarquía clases “guiadas por las responsabilidades”, es decir, que pretenden satisfacer todas las posibilidades y de esta forma las nuevas funcionalidades se componen de piezas simples que se crean y se combinan con facilidad, independientemente de los objetos cuyo comportamiento extienden.

En Symfony2 este patrón es fácilmente visible ya que la vista se separa por niveles, en hasta 3 niveles, una plantilla base y varias plantillas que heredan de esta. Normalmente, la plantilla base es global en toda la aplicación y contiene el código HTML que es común a la mayoría de las páginas. Su uso aporta una mayor flexibilidad que la herencia estática, permitiendo, entre otras cosas, añadir una funcionalidad varias veces.

2.9 Conclusiones del capítulo

Luego de definir las características que debe cumplir el sistema para la administración centralizada de dispositivos del hardware y de realizar el análisis y diseño correspondiente, se concluye que durante el desarrollo del capítulo se precisaron las diferentes historias de usuario definiendo así un plan de duración y un plan de entrega para la realización del sistema, permitiendo que el mismo cumpla con las funcionalidades requeridas y con el tiempo estimado. Además se identificaron las personas que intervendrán en el sistema y se seleccionó la arquitectura n-capas y el estilo arquitectónico modelo-vista-controlador que garantizará el desarrollo de la aplicación con el uso apropiado de los patrones de diseño para lograr un software con calidad.

Capítulo 3 Implementación y prueba

Introducción

La implementación en el proceso de desarrollo de un software adquiere gran importancia debido a que le da funcionalidad al producto que se desarrolla. Además, son importantes las pruebas que se le realizan al mismo para validar su correcto funcionamiento. El capítulo abarca las fases de implementación y prueba de la aplicación. Se exponen las tareas asignadas a las HU para llevar a cabo la implementación. Se muestra el estándar de codificación utilizado y el código de las HU con mayor prioridad de desarrollo. Se describen las pruebas realizadas a la aplicación con el objetivo de verificar si se cumplieron los requerimientos de la misma.

3.1 Fase de implementación

En la fase de Planificación se detallaron las HU correspondientes a cada una de las iteraciones a desarrollar. Estas HU se descomponen en tareas de programación o ingeniería, que se asignan a un equipo de desarrollo o persona. Estas tareas no tienen que necesariamente ser entendidas por el cliente, pueden ser escritas en lenguaje técnico y son para el uso estricto de los programadores.

A continuación se detallan cada una de las iteraciones y tareas de ingeniería asignadas a cada HU deseadas.

Iteración 1

En esta iteración se desarrollan las HU pertenecientes al módulo: Gestionar recursos, dando al sistema las primeras funcionalidades, estas son: gestionar agrupaciones lógicas, mostrar los datos de una agrupación lógica dado su identificador, gestionar agrupaciones físicas, gestionar tipos de dispositivos, gestionar dispositivos y gestionar servicios. Las restantes tareas de ingeniería de la iteración 1 podrán ser consultadas en el Anexo 3.

- HU1 Gestionar agrupaciones lógicas

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Diseño de la interfaz para gestionar agrupaciones lógicas.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**0.25
Fecha Inicio: 19/1/2015	**Fecha Fin: 20/1/2015**
Programador Responsable: Lien Morell Baró/Yasiel Lora Rodríguez	
Descripción: Se implementa el diseño de la interfaz para gestionar agrupaciones lógicas con los componentes necesarios para su funcionamiento.	

Tabla 8 Tarea de Ingeniería #1 HU gestionar agrupaciones lógicas

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Conexión con la base de datos	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**0.25
Fecha Inicio:21/1/2015	**Fecha Fin:** 22/1/2015
Programador Responsable: Lien Morell Baró/Yasiel Lora Rodríguez	
Descripción: Se realiza la conexión con la base de datos definida para la aplicación.	

Tabla 9 Tarea de Ingeniería #2 HU gestionar agrupaciones lógicas

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 1
Nombre Tarea: Prueba de la funcionalidad agrupaciones lógicas	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**0.50
Fecha Inicio: 22/1/2015	**Fecha Fin:** 23/1/2015
Programador Responsable: Lien Morell Baró/Yasiel Lora Rodríguez	
Descripción: Se le realizan una serie de pruebas funcionales a la aplicación, con la finalidad de obtener posibles fallas al realizar esta tarea.	

Tabla 10 Tarea de Ingeniería #3 HU gestionar agrupaciones lógicas

Iteración 2

En esta iteración se desarrollan las HU pertenecientes al módulo: Estaciones de trabajo, las cuales permitirán activar estación de trabajo, desactivar estación de trabajo, asignar servicios a las estaciones de

trabajo según la ubicación lógica y asignar dispositivos a las estaciones de trabajo según la ubicación lógica. El resto de las tareas de ingeniería se encuentran en el Anexo 3.

- HU7 Activar estación de trabajo

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 7
Nombre Tarea: Diseño de la interfaz activar estación de trabajo	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.25
Fecha Inicio: 9/3/2015	**Fecha Fin:** 10/3/2015
Programador Responsable: Lien Morell Baró/Yasiel Lora Rodríguez	
Descripción: Se implementa el diseño de la interfaz para para activar estación de trabajo con los componentes necesarios para su funcionamiento.	

Tabla 11 Tarea de Ingeniería #1 HU Activar estación de trabajo

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 7
Nombre Tarea: Conexión con la base de datos	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**0.25
Fecha Inicio:11/3/2015	**Fecha Fin:** 12/3/2015
Programador Responsable: Lien Morell Baró/Yasiel Lora Rodríguez	
Descripción: Se realiza la conexión con la base de datos definida para la aplicación.	

Tabla 12 Tarea de Ingeniería #2 HU Activar estación de trabajo

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 7
Nombre Tarea: Prueba de la funcionalidad activar estación de trabajo	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**0.50
Fecha Inicio: 12/3/2015	**Fecha Fin:** 13/3/2015
Programador Responsable: Lien Morell Baró/Yasiel Lora Rodríguez	
Descripción: Se le realizan una serie de pruebas funcionales a la aplicación, con la finalidad de obtener posibles fallas al realizar esta tarea.	

Tabla 13 Tarea de Ingeniería #3 HU Activar estación de trabajo

Iteración 3

En esta iteración se desarrollan las HU pertenecientes al módulo: Módulo Reportes, las cuales permitirán generar reportes de las de estaciones de trabajo activadas y conocer estado del ordenador (encendido/apagado). El resto aparecen en Anexo 3.

- HU11 Generar reportes

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 11
Nombre Tarea: Diseño de la funcionalidad generar reportes	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.50
Fecha Inicio: 6/4/2015	**Fecha Fin:** 8/4/2015
Programador Responsable: Lien Morell Baró/Yasiel Lora Rodríguez	
Descripción: Se implementa el diseño de la interfaz para para generar los reportes de uso con los componentes necesarios para su funcionamiento.	

Tabla 14 Tarea de Ingeniería #1 HU11 Generar reportes

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 11
Nombre Tarea: Conexión con la base de datos	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**0.50
Fecha Inicio:8/4/2015	**Fecha Fin:** 10/4/2015
Programador Responsable: Lien Morell Baró/Yasiel Lora Rodríguez	
Descripción: Se realiza la conexión con la base de datos definida para la aplicación.	

Tabla 15 Tarea de Ingeniería #2 HU11 Generar reportes

3.2 Estándar de codificación

Establecer un estándar de codificación que sea aceptado por todo el equipo de desarrollo es muy importante en una metodología como XP que promueve la propiedad colectiva del código y la constante refactorización. El propósito fundamental de los estándares de codificación es que el sistema en cuestión tenga una arquitectura y un estilo consistente, con lo cual resulte fácil de entender y por supuesto fácil de mantener.

Los estándares de codificación son un complemento a la programación por pares, o sea en equipo, y no sólo es importante usar un estándar, sino usar un buen estándar de codificación, de esta forma se deberá promover la intención del código e incorporar las mejores prácticas de la codificación.

Entre otros aspectos se apunta a las variaciones en las convenciones o adiciones necesarias en el código del lenguaje PHP especialmente dirigido al trabajo con el marco de trabajo Symfony2. Los principales aspectos incluidos en dicho estándar se relacionan a continuación:

- Agregar un único espacio alrededor de operadores (==, &&).
- Agregar un único espacio antes de los paréntesis de apertura de una palabra clave de control (if, else, for, while).
- Agregar una línea en blanco antes de la sentencia return.
- Utiliza constantes de tipo PHP nativas en minúsculas: false, true y null. Lo mismo aplica para array ().
- Utilizar llaves para indicar el cuerpo de las estructuras de control sin importar el número de sentencias que éstas contengan.
- Colocar las llaves en sus propias líneas para clases, métodos y declaración de funciones.
- Definir una clase por archivo.
- Declarar las propiedades de las clases antes de los métodos.
- Declarar los métodos públicos primero, luego los protegidos y finalmente los privados.

Convención de Nombres

- Utilizar camelCase y no guiones bajos, para las clases, variables, funciones y nombres de métodos.

- Utilizar guiones bajos para definir opciones, argumentos y nombres de parámetros.
- Utilizar los namespace para todas las clases.
- Utilizar Symfony como el namespace de primer nivel.

3.3 Diagrama de despliegue

El diagrama de despliegue describe la distribución física de un software en un ambiente de producción o prueba. En él se grafican los nodos y las conexiones necesarias para el funcionamiento exitoso del producto software. Muestra también, dónde se localizan los componentes del software, o sea, en qué servidores, ordenadores o dispositivo hardware. La figura 6 muestra el diagrama de despliegue correspondiente al sistema de administración centralizada de dispositivos del hardware. Este está compuesto por dos servidores, el primero de ellos será el Servidor de Aplicaciones y contendrá el Sistema para el control centralizado de dispositivos. El segundo servidor contendrá la base de datos del sistema y deberá poseer PostgresSQL. La comunicación entre dichos servidores será a través de la clase pgp_conection.

El diagrama representa además una estación cliente en cual permitirá la interacción con los dispositivos. Esta estación de trabajo utiliza el protocolo SOAP[1] para la comunicación con el servidor de aplicaciones.

[1] (*Simple Object Access Protocol) Protocolo para el intercambio de mensajes utilizando el formato XML*

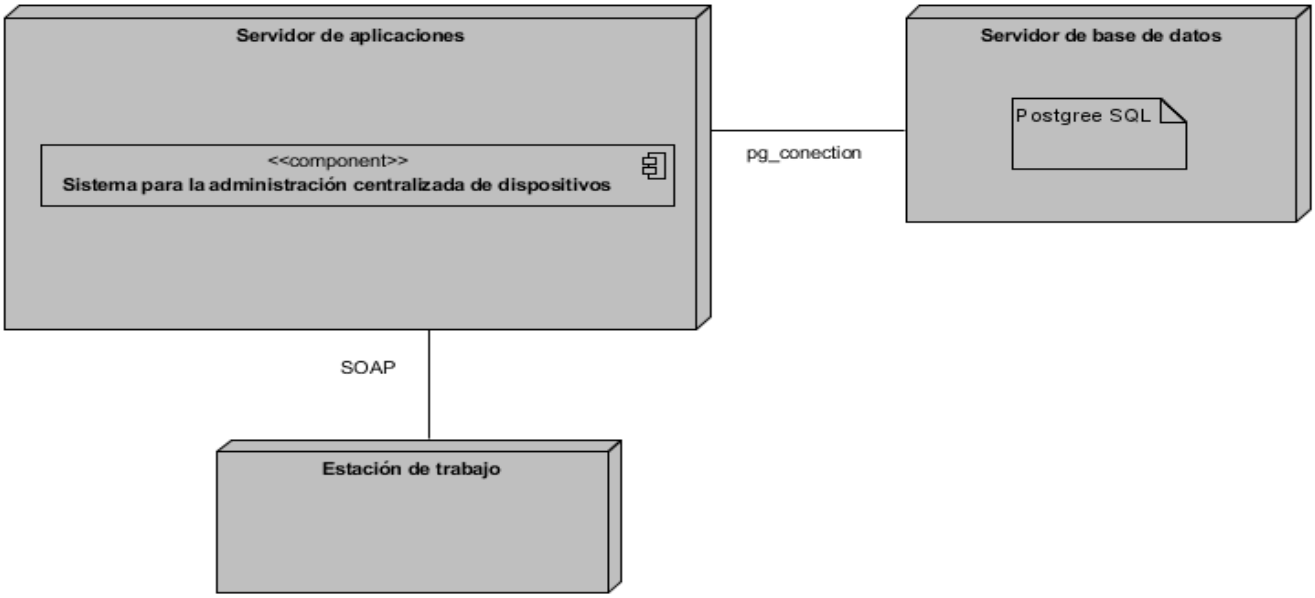

Figura 6 Diagrama de despliegue

3.4 Fase de prueba

Una de las características fundamentales que posee la metodología XP es el proceso de prueba para comprobar las funcionalidades a medida que estas son implementadas, y a su vez mostrar al cliente los resultados obtenidos durante el desarrollo del producto. De esta forma se busca disminuir la cantidad de errores no detectados reduciendo el tiempo transcurrido entre la aparición de un error y su detección, aumentando así la calidad de la aplicación informática y la seguridad de evitar efectos colaterales a la hora de realizar modificaciones.

Esta metodología ágil divide las pruebas en dos grupos: pruebas unitarias y pruebas de aceptación. Las pruebas unitarias tienen como objetivo revisar el código de forma automática y son desarrolladas por los programadores. En cambio las pruebas de aceptación están destinadas a verificar que las historias de usuario cumplen con las funcionalidades requeridas por el cliente al final de cada iteración. Las pruebas de aceptación tienen mayor peso que las unitarias ya que representan un indicador de la satisfacción del

cliente con el producto. Este tipo de pruebas debe realizarse lo más rápido posible para que los programadores puedan hacer los cambios que se necesiten con mayor rapidez.

3.4.1 Pruebas unitarias

En la programación, una prueba unitaria o de unidad es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada una de las partes que integran la aplicación funcione correctamente por separado. (44)

Las pruebas unitarias se realizan para controlar el funcionamiento de pequeñas porciones de código como son subprogramas o métodos (en la programación orientada a objeto POO). Generalmente son realizadas por los mismos programadores puesto que al conocer con mayor detalle el código, se les simplifica la tarea de elaborar conjuntos de datos de prueba para probarlo.

PHP permite realizar pruebas a través del entorno PHPUnit a través de consola, a continuación se muestran las pruebas unitarias realizadas a la funcionalidad gestionar Agrupación lógica pertenecientes al módulo Gestionar recursos.

Figura 7 Prueba unitaria: Gestionar agrupación lógica

3.4.2 Pruebas de aceptación

Las pruebas de aceptación se realizan para asegurar que las funcionalidades cumplen su objetivo y que la aplicación informática es lo que el cliente realmente necesita. Estas pruebas son creadas a partir de las historias de usuario y es el cliente quien especifica los aspectos para comprobar que se han implementado correctamente. Una historia de usuario puede tener todas las pruebas de aceptación que necesite para garantizar su buen funcionamiento. Estas pruebas funcionan como una caja negra ya que cada una de ellas representa una salida esperada de la aplicación informática. Los clientes son los responsables de verificar que los resultados de las pruebas sean correctos y de tomar decisiones acerca de las mismas. Una vez que todas las historias de usuario hayan pasado su prueba de aceptación, entonces se considera terminada la aplicación informática.

3.4.2.1. Pruebas de caja negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa. O sea, a través de los casos de prueba se demuestra que las funciones del software son operativas, que la entrada sea acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. (45) Se realizaron pruebas a diferentes funcionalidades del sistema una de ellas se muestra a continuación y las restantes se pueden observar en el Anexo 4.

Nombre del Flujo	Escenarios del flujo	Descripción de la funcionalidad	Flujo central
1.Gestionar agrupaciones lógicas	1.1 Agregar agrupación lógica correctamente.	El sistema agrega la agrupación lógica.	1. Gestionar 2.Agrupación lógica 3. Adicionar 4. Aceptar

	1.2 Agregar agrupación lógica con campos incorrectos.	El sistema muestra un mensaje de alerta indicando que existe un campo incorrecto.	1. Gestionar 2.Agrupación lógica 3. Adicionar 4. Aceptar
	1.3 Agregar agrupación lógica con campos vacíos.	El sistema muestra un mensaje de alerta indicando que existe un campo vacío.	1. Gestionar 2.Agrupación lógica 3. Adicionar 4. Aceptar
	1.4 Modificar agrupación lógica correctamente.	El sistema modifica la agrupación lógica.	1. Gestionar 2. Agrupación lógica 3. Modificar 4. Aceptar 5. Si
	1.5 Modificar agrupación lógica con campos incorrectos.	El sistema muestra un mensaje de alerta indicando que existe un campo incorrecto.	1. Gestionar 2.Agrupación lógica 3. Adicionar 4. Aceptar
	1.6 Modificar agrupación lógica con campos vacíos.	El sistema muestra un mensaje de alerta indicando que existe un campo vacío.	1. Gestionar 2.Agrupación lógica 3. Adicionar 4. Aceptar
	1.7 Modificar agrupación	El sistema muestra un mensaje	1. Gestionar 2.Agrupación lógica

	lógica sin haber seleccionado la agrupación a modificar.	indicando que debe seleccionar una agrupación.	3. Adicionar 4. Aceptar
	1.8 Eliminar agrupación lógica correctamente	El sistema elimina la agrupación lógica.	1. Gestionar 2. Agrupación lógica 3. Eliminar 4. Aceptar 5. Si
	1.9 Eliminar agrupación lógica sin haber seleccionado la agrupación a eliminar.	El sistema muestra un mensaje indicando que debe seleccionar una agrupación.	1. Gestionar 2. Agrupación lógica 3. Eliminar 4. Aceptar
	1.10 Cancelar la gestión de agrupaciones lógicas	El sistema re direcciona a la página principal.	1. Gestionar 2. Agrupación lógica 3. Cancelar

Tabla 16 Caso de prueba: Gestionar agrupaciones lógicas. Fuente: Elaboración propia.

ID del Escenario	Tipos de Clases	Clases	Resultado Esperado	Resultado de la Prueba	Observaciones
1.1	**Clases Válidas**	**Nombre:** Sección Sindical CISED **Tipo de agrupación lógica**: Oficina	Se agrega una nueva agrupación lógica	Se agrega una nueva agrupación lógica	
1.2		**Nombre:**	El sistema	El sistema	

	Clases Inválidas	Sección Sindical CISED **Tipo de agrupación lógica**: 087*	muestra un mensaje de alerta indicando que existe un campo incorrecto.	muestra un mensaje de alerta indicando que existe un campo incorrecto.	
1.3	**Clases Inválidas**	**Nombre: Tipo de agrupación lógica:** Oficina	Se muestra un mensaje que indica que el campo está vacío.	Se muestra un mensaje que indica que el campo está vacío.	
1.4	**Clases Válidas**	**Nombre:** "Camagüey 2" **Tipo de agrupación lógica:** Impresión	Se modifica la agrupación lógica.	Se modifica la agrupación lógica.	
1.8	**Clases Válidas**	**Agrupación lógica:** "Camagüey 2"	El sistema elimina la agrupación lógica.	El sistema elimina la agrupación lógica.	

Tabla 17 Iteraciones. Fuente: Elaboración propia.

3.5 Resultados de las pruebas

Durante la ejecución de tres iteraciones de pruebas se detectaron un conjunto de diez no conformidades, las mismas fueron mitigadas. En la primera iteración se realizó una versión funcional del software que fue probado en el entorno de desarrollo del Proyecto Identidad Cuba con el objetivo de probar el sistema y corregir cualquier deficiencia antes de ser desplegado en un ambiente real.

En dicha iteración se encontraron un total de 7 no conformidades, que en todos los casos fueron problemas de validación de datos relacionados con la entrada de caracteres no válidos en campos de texto de las interfaces. Luego de solucionar las dificultades detectadas se realizó una segunda iteración generando una versión funcional del producto. En esa iteración se comprobó que no quedaban errores de la primera, pero se encontraron 3 nuevas no conformidades relacionadas con la integración del servicio local del DGM V3.0 que fueron solucionados y culminando con una exitosa prueba en la tercera iteración dándole al sistema un nivel de seguridad mayor.

Figura 8 Resultados de las pruebas

3.6 Conclusiones parciales

En este capítulo se definió el estilo de codificación utilizado en la implementación del sistema, lo que permitió mantener una uniformidad en la codificación y mayor claridad a la hora de leer o agregar líneas de código. El modelo de despliegue definido mostró con mayor claridad la distribución física del sistema sobre una arquitectura de hardware. Luego de la realización de pruebas unitarias, se comprobó el correcto funcionamiento de procedimientos internos del software. A partir del empleo de estos tipos de pruebas se encontraron un conjunto de no conformidades, las cuales fueron la base fundamental para que el sistema mejorara su rendimiento y funcionamiento. Además las pruebas realizadas brindaron un resultado satisfactorio que le aporta al sistema mayor nivel de robustez y validez.

Conclusiones generales

A partir del desarrollo de la presente investigación se arribaron a las siguientes conclusiones:

- El análisis de los sistemas existentes tanto en Cuba como en el mundo dio como resultado la necesidad real de desarrollar un sistema que permita la administración con dispositivos del hardware que interactúan con aplicaciones web en ambientes multiplataforma.
- Se realizó un estudio de las herramientas y tecnologías necesarias para el desarrollo del sistema, definiendo así los lenguajes, herramientas y métodos a seguir para su implementación.
- La descripción de escenarios presentes en la fase de planificación definida por la metodología XP, permitió el conocimiento de las funcionalidades del sistema.
- Se realizó el plan de iteraciones que permitió la organización del tiempo y el trabajo con el fin de culminar el desarrollo del sistema en el tiempo requerido.
- Las pruebas realizadas permitieron comprobar las funcionalidades del sistema, estas brindaron resultados satisfactorios otorgando validez a la investigación.

Recomendaciones

Para mejorar el sistema se recomienda:

- Implementar otras versiones de este Subsistema incorporando funcionalidades nuevas que sean útiles para lograr un mejor producto.
- Presentar la investigación en eventos de cortes científicos.
- Realizar la integración con las demás aplicaciones que conformarán el Device Grid Manager.

Referencias bibliográficas

1. Manual de Usuario. Device Grid ManagerV3.0. 2012 : s.n.

2. Definicion.de. [En línea] 2010. [Citado el: 13 de noviembre de 2015.] http://definicion.de/dispositivo/.

3. axiompass. axiompass. [En línea] 2010. [Citado el: 9 de enero de 2015.] http://axiompass.com/Solutions/Access.aspx.

4. Inform@tizando. [En línea] 2010. [Citado el: 15 de diciembre de 2015.] http://informaticandotf.blogspot.com/2010_08_01_archive.html.

5. El sistema operativo GNU. [En línea] 2002. [Citado el: 15 de febrero de 2015.] http://www.gnu.org/philosophy/free-sw.es.html.

6. Letelier, Patricio, Canós, José H. & Penadés, Mª Carmen. Metodologías ágiles en el desarrolo de software:Extreme Programming (XP). 2010.

7. García, Félix Óscar Rubio y Santos, Crescencio Bravo. Escuela de Ingeniería Civil Informática [Online] 2009-2010. [En línea] 2009-2010. [Citado el: 10 de noviembre de 2014.] http://www.eici.ucm.cl/Academicos/R_Villarroel/descargas/ing_sw_1/Metodologias.pdf..

8. Figueroa , Roberth, Solis, Camilo y Cabrera, Armando. Metodologías tradicionales y metodologías ágiles. 2010.

9. Sánchez Asenjo, Jorge. Apuntes Completos sobre Sistemas gestores de Bases de Datos. [En línea] 2009. [Citado el: 17 de Febrero de 2015.] http://ubuntuone.com/p/sqt/ubuntuone.pdf.

10. Flores, Lic. Ervin. Metodología agiles Proceso Unificado Ágil (AUP). 2006.

11. Schwaber, K. Advanced Development Methods. SCRUM Development. 2010.

12. Kniberg, Henrik. Scrum y XP desde las trincheras.pdf. 2009.

13. Kenneth E. Kendall, Julie E. Kendall. Análisis y diseño de sistemas. 2005.

14. Paradigm, Visual. Visual Paradigm. Visual Paradigm. [En línea] 2012. [Citado el: 11 de noviembre de 2014.] http://www.visual-paradigm.com./..

15. Louden, Kenneth C. Lenguajes de programación: principios y práctica. 2004.

16. Mehdi Achour, Friedhelm Betz y Antony Dovgal . http://php.net/manual/es/index.php. [En línea] 2010. [Citado el: 12 de marzo de 2015.]

17. Alvarez, M. A. Desarrollo Web. 2007.

18. Clemente, C. Python Software de Comunicación." Volumen, 50 DOI:. 2007.

19. Eguiluz, J. Desarrollo web ágil con Symfony2. 2011.

20. Zend Technologies Ltd. [En línea] 2006 – 2013. [Citado el: 13 de marzo de 2015.] http://framework.zend.com/about/.

21. Winesett, Jeffrey. Agile Web Application Development with Yii1.1 and PHP5. 2010.

22. Seoane Pascual, Joaquín, Jesús M. , González Barahona y Robles, Gregorio . [En línea] 2003-2007. [Citado el: 11 de noviembre de 2014.] http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/sec-.

23. NetBeans. NetBeans. Oracle Corporation and/or its affiliates, 2012. [En línea] 2012. [Citado el: 12 de noviembre de 2014.] http://netbeans.org/..

24. IBM. [En línea] [Citado el: 12 de mayo de 2015.] http://www.ibm.com/developerworks/ssa/library/os-ecov/.

25. Zend the PHP Company. [En línea] [Citado el: 20 de febrero de 2015.] http://www.framework.zend.com/.

26. Foundation, The Apache Software. Documentación del Servidor HTTP Apache 2.4. [En línea] 2011. [Citado el: 10 de noviembre de 2014.] http://httpd.apache.org/docs/2.4/es/.

27. fraterneo GNU/Linux.Promoviendo el Software Libre y la Educación... [En línea] 12 de julio de 2010. http://fraterneo.blogspot.com/2010/12/6-servidores-web-libres-para-gnulinux-y.html.

28. Igor Sysoev. nginx. [En línea] 4 de octubre de 2004. [Citado el: 23 de noviembre de 2014.] http://nginx.org.

29. Marquez, Maria y Marquez, Andres. Apuntes de ficheros y bases de datos. 2012. 2012.

30. DEMO E-DUCATIVA CATEDU. [En línea] 2010. [Citado el: 31 de marzo de 2015.] http://e-ducativa.catedu.es/44700165/aula/archivos/repositorio/1000/1080/html/31_clasificacin_de_los_sgbd.html.

31. Group, PostgreSQL Global Development. PostgreSQL 8.4.1 Documentation. California. pag. 2119 : s.n., 2011.

32. Medrano, Jesús Rafael Sánchez. Introducción a Base de Datos con PostgreSQL. 2011.

33. Espinoza., Humberto. PostgreSQL: Una Alternativa de DBMS Open Source. PostgreSQL: Una Alternativa de DBMS Open Source. [En línea] 2011. [Citado el: 2 de octubre de 2014.] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf..

34. CORPORATION, Oracle. Oracle. [En línea] 2010. [Citado el: 1 de octubre de 2014.] http://www.oracle.com/global/es/index.html..

35. mysql.The world's most popular open source database. [En línea] [Citado el: 13 de noviembre de 2014.] http://www.mysql.com/.

36. Corporation, MySQL. MySQL. MySQL.org . [En línea] 2012. [Citado el: 1 de octubre de 2014.] http://www.postgresql.org/docs/8.3/static/ddl..

37. programacion.net. [En línea] [Citado el: 4 de octubre de 2014.] programacion.net. [En línea] [Citado el: 4 de octubre de 2 http://programacion.net/articulo/conceptos_basicos_de_orm_object_relational_mapping_349..

38. Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. La Habana : Felix Varela, 2004.

39. Beck. Extreme Programming Explained. 2009.

40. R Jeffries, A Anderson. Extreme Programming Installed. s.l. : Addison-Wesley, 2006.

41. Pressman, Roger S. Ingenieria del Software. Un enfoque práctico .Parte IV.

42. Pressman, Roger S. INGENIERÍA DE SOFTWARE. Un enfoque práctico. 2005.

43. Carlos A Guerrero, Johanna M Suárez. scielo. scielo. [En línea] 2013. [Citado el: 3 de mayo de 2015.] ttp://www.scielo.cl/scielo.php?.

44. B, CARLOS. QUnit, testeando nuestras aplicaciones. [En línea] 2011. [Citado el: 10 de febrero de 2015.] http://www.etnassoft.com/2011/02/01/qunit-testeando-nuestras-aplicaciones-javascript/.

45. Manuel, T,G. Técnicas de Prueba. 2012.

46. Sánchez, María A. Mendoza. Metodologías del desarrollo de software. 2009.

47. Apache Software Foundation. [En línea] httpd.apache.org.

48. globetesting. [En línea] 2011. [Citado el: 22 de abril de 2015.] http://www.globetesting.com/pruebas-de-rendimiento/.

49. Maestros de la web. [En línea] 15 de febrero de 2013. [Citado el: 28 de marzo de 2015.] http://www.maestrosdelweb.com/curso-symfony2-introduccion-instalacion/.

50. [En línea] [Citado el: 9 de enero de 2015.] Codejobs.biz\Estándares de codificación en PHP - Aprende a Programar.

51. Baluarte.NET. [En línea] 2008. [Citado el: 12 de marzo de 2015.] http://www.baluart.net/search/node/Introducci%C3%B3n%20a%20las%20pruebas%20unitarias%20en%20PHP%20con%20PHPUnit.

52. Manual de Symfony2 en Español. [En línea] 2011. [Citado el: 21 de abril de 2015.] Manual de Symfony2 en Español.com.

53. Alc@soft.es. [En línea] [Citado el: 12 de noviembre de 2014.] Alc@soft.es/Pruebas unitarias/Alcasoft Java pruebas unitarias con JUnit y NetBeans.

54. Symfonyando. [En línea] [Citado el: 13 de noviembre de 2014.] http://www.leccionespracticas.com/php/instalacion-de-symfony2-en-wamp-con-php5-3-extension-intl-y-acelerador-acp-resuelto/.

55. Cristablab. [En línea] 2009. [Citado el: 13 de octubre de 2014.] http://www.cristalab.com/tutoriales/reglas-de-codificacion-y-lineamientos-de-codigo-php-c190l/.

56. Pruebas de control de flujo de datos. [En línea] 2012. [Citado el: 20 de septiembre de 2014.] http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf.

57. Pruebas de caja blanca automatizadas para la plataforma.NET. [En línea] 2011. [Citado el: 15 de noviembre de 2014.] http://research.microsoft.com/en-us/projects/pex/.

58. Recopilación linux. [En línea] [Citado el: 21 de septiembre de 2014.] http://yuriang85.cubava.cu/2015/04/30/dia-8-pruebas-unitarias/.

59. Escuela de Ingeniería de Sistemas y Computación. Escuela de Ingeniería de Sistemas y Computación. [En línea] [Citado el: 1 de mayo de 2015.] http://eisc.univalle.edu.co/cursos/web/material/750073/1/GSQA-IntroduccionPruebasSoftware.ppt..

60. MiTecnologico. MiTecnologico. [En línea] 2012. [Citado el: 12 de marzo de 2014.] http://www.MiTecnologico.com.

61. Castro, Elizabeth. HTML con XHTML y CSS. Madrid : Anaya Multimedia : s.n., 2009. p. 656. ISBN 9788441521834.

62. Barcia, Diego. Maestros de la Web. ¿Qué es CSS? [En línea] [Citado el: 6 de marzo de 2015.] http://www.maestrosdelweb.com/editorial/introcss/.

63. Foundation, Mozilla. Mozilla Addons. Mozilla Addons. [En línea] 16 de marzo de 2010. [Citado el: 14 de diciembre de 2014.] https://addons.mozilla.org/..

64. sagem. sagem. [En línea] 15 de marzo de 2010. [Citado el: 10 de noviembre de 2014.] http://www.sagem.com/index.php?id=774&L=8..

65. ALTOVA. Altova MissionKit. [En línea] 2011. [Citado el: 2 de diciembre de 2014.] http://www.altova.com/es/missionkit/software-.

66. Marley, Jimi. ¿Por qué elegir PHP? [En línea] 2011. [Citado el: 5 de noviembre de 2014.] http://www.programacion.com/articulo/por_que_elegir_php_143.

67. Martínez, Rafael. Sobre PostgreSQL. [En línea] 2010. [Citado el: 10 de marzo de 2015.] http://www.postgresql.org.es/sobrepostgresql#caracteristicas.

68. PELAEZ, J. C. Arquitectura basada en capas . [En línea] [Citado el: 2 de abril de 2015.] http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx..

69. BANNISTER, T. What is Apache? [En línea] 17 de enero de 2014. http://wiki.apache.org/httpd/FAQ#What_is_Apache.3F.

70. BERGMANN, S. PHPUnit github. [En línea] [Citado el: 28 de marzo de 2015.] https://github.com/sebastianbergmann/phpunit/.

71. Duque, Raúl González. Python para todos. [En línea] 2010. [Citado el: 9 de febrero de 2014.] http://www.easyeclipse.org/site/home/.

72. CORPORATION, M. Trabajar con pruebas unitarias. [En línea] [Citado el: 2 de mayo de 2015.] http://msdn.microsoft.com/es-es/library/ms182515%28VS.80%29.aspx.